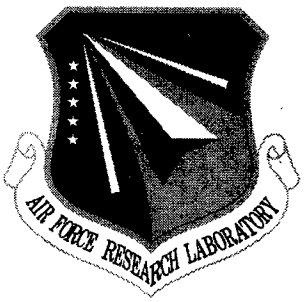
AFRL-IF-RS-TR-2000-1
Final Technical Report
January 2000



METAMODELING TECHNIQUES AND APPLICATIONS

University of Colorado at Colorado Springs

Don Caughlin

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2000-1 has been reviewed and is approved for publication.

APPROVED: 

ALEX F. SISTI
Project Engineer

FOR THE DIRECTOR: 

JAMES W. CUSACK, Chief
Information Systems Division

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFSB, 525 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	JANUARY 2000	Final Mar 96 - Sep 98

4. TITLE AND SUBTITLE
METAMODELING TECHNIQUES AND APPLICATIONS

5. FUNDING NUMBERS
C - F30602-96-C-0040
PE - 62702F
PR - 4594
TA - 15
WU - N6

6. AUTHOR(S)
Donald Caughlin

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Colorado at Colorado Springs
PO Box 7150
1420 Austin Bluffs Parkway
Colorado Springs CO 80933-7150

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFSB
525 Brooks Road
Rome NY 13441-4514

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2000-1

11. SUPPLEMENTARY NOTES
Air Force Research Laboratory Project Engineer: Alex F. Sisti/IFSB/(315) 330-3983

12a. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
This report represents a compendium of previously published reports generated during the conduct of this contract. They describe model abstraction techniques in general; reduced order metamodeling as a specific abstraction technique, and other applications of metamodeling.

14. SUBJECT TERMS
Metamodeling, Reduced Order Modeling, Model Abstraction, System Identification Techniques, VV&A

15. NUMBER OF PAGES
88

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Standard Form 298 (Rev. 2-89) (EG)
Prescribed by ANSI Std. 239.18
Designed using Perform Pro, WHS/DIOR, Oct 94

Contents

Acknowledgement

We are most grateful to the Air Force, and in particular to both Al Sisti and Steve Farr of the Air Force Research Laboratory/Information Directorate (formerly Rome Laboratory). Our research would not be possible without their financial support. The topic of this contract was development of a Metamodeling Support System (MSS) to assist in the metamodeling of complex simulations. The research resulted in a complete set of specifications: Metamodeling Support System Requirements from Final Report F30602-94-0110; Metamodeling Support System System/Subsystem Specification (SSS); Metamodeling Support System System/Subsystem Design Description (S/SDD); Metamodeling Support System Software Design Description (SDD). In addition to the specifications, the research also yielded a prototype GUI and a database structure to support the MSS. Also, the research resulted in several publications and a significant amount of software engineering relating to the C++ - Expert System interface.

Don Caughlin

Introduction

This report represents a compendium of previously published reports generated during the conduct of Air Force Research Laboratory contract F30602-96-C-0040. They describe the use of Metamodeling as a model abstraction technique, provide a process and procedures to efficiently generate metamodels, and provide the structure of a software system that can support the construction of metamodels.

Alex F. Sisti
AFRL/IFSB
December 1999

Model Abstraction Via Solution of A General Inverse Problem to Define a Metamodel

Don Caughlin

Space and Flight Systems Laboratory
University of Colorado at Colorado Springs
Colorado Springs, Colorado, 80918
donc@mozart.uccs.edu

February 13, 1997

Abstract

Historically, most metamodels were generated by linear regression to determine the best polynomial fit to a set of input-output data. This paper presents a new formulation for definition of the metamodeling problem, a new framework for the solution, and a structured method to attain that solution. Most importantly, model abstraction via solution of a general inverse problem expands the available classes of metamodels by supporting the development of dynamical models that incorporate memory. This expansion allows the generation of metamodels that include system dynamics so that metamodels can be developed where the past can influence the future. Defining the metamodeling problem in this manner adds a vast amount of existing research (realization theory and system identification methods) to the statistical (regression) methods currently used to approach the problem. This framework has been successfully applied to a number of metamodeling problems and is applicable in all areas of Modeling and Simulation. It can be used to support simulation analysis by reducing simulation results to a set of mathematical equations that can be easily analyzed. This technology can also be used in the integration of multiple simulations by approximating one of the simulations (or portions thereof) with a system of equations. It can also be used in the Verification, Validation and Accreditation (VV&A) process by providing a direct, external and efficient comparison of different models or simulations. [1]

[1] This work was supported in part by The USAF Rome Laboratory Contract F30602-94-C-0110

1 Introduction

Large simulations, like the tactical simulation models used by the Department of Defense to assess the capabilities of combat systems and tactics, are highly complex. While these simulations can provide specific information, it is often difficult to determine the relationship of individual factors to the performance of the modeled process [1]. Consequently, it is not easy to use the results of the model in another simulation or couple multiple models to investigate a larger issue. The result is a proliferation of point designed models and simulations, expensive upgrade and maintenance and the inability to efficiently answer many of the more difficult questions raised by decision makers [2].

A technique called "metamodeling" offers the ability to facilitate this type of assessment. A metamodel is a mathematical approximation of the system relationships defined by a high fidelity model or simulation. As an abstraction, a metamodel is a projection of the model onto a subspace defined by new constraints or regions of interest.

Historically, most metamodels were generated by linear regression to determine the best polynomial fit to a set of input-output data. In this paper we present a new approach where in we do not try to fit data but concentrate on the identification of the underlying systems that defined the process. The focus is not on statistics but on the system theoretic properties of the manifest behavior.

By focusing on the system theoretic properties of the manifest behavior, we generate the metamodel via solution of a general inverse problem and do not restrict the solution to an approximation of the input-output map. This approach expands the available

classes of metamodels by supporting the development of dynamical models that incorporate memory. This expansion allows the generation of metamodels that include system dynamics so metamodels can be developed that allow the past to influence the future. Defining the metamodeling problem in this manner adds a vast amount of existing research (realization theory and system identification methods) to the statistical (regression) methods currently used to approach the problem.

In addition to a new approach to the definition of the problem we present a new framework for the solution [3, 4]. The framework centers on the behavior of the system, the behavioral equations that specify the behavior and latent variables which may be present from first principles. The theory, structure and definitions follow the presentation given in [5] and begins with the essence of the system and not with a structure and assumptions that facilitate a solution technique. Consequently, this theory provides a basis that includes all of the issues associated with modeling and modeling from data.

Although the new framework was consistent with existing metamodeling procedures defined in [1], the development of the metamodel required too many decisions involving: *a priori* knowledge; the data; possible metamodel sets; and rules to determine the best model set to realize the data. Each decision was a complex function of *a priori* information and prior selections in the metamodeling process.

A structured metamodeling method is presented that addresses this complexity. The structure is based on the fact that the construction of a metamodel (selection of the parameters used for the projection) is determined by the metamodel set, method of identification and identification criteria. The method is based on a new taxonomy of metamodel sets and identification methods that allows the separation of the metamodeling process into a set of sequential decisions based on *a priori* information.

This framework has been successfully applied to a number of metamodeling problems and is applicable in all areas of Modeling and Simulation [3, 4]. It can be used to support simulation analysis by reducing simulation results to a set of mathematical equations that can be easily analyzed. This technology can also be used in the integration of multiple simulations by approximating one of the simulations (or portions thereof) by a system of equations. It can also be used in the Verification, Validation and Accreditation (VV&A) process by providing a direct, external, and efficient comparison of different models or simulations [6].

This paper provides the approach, framework, and metamodeling method. While the paper also presents possible model sets and identification methods that can be used with these definitions, it is beyond the scope of the paper to present all of the details associated with these sets and methods. Additional information on these latter subjects can be found in the references.

The paper is organized as follows: Section 2 introduces metamodels, and metamodeling via direct and inverse modeling. Section 3 presents the identification framework discussing dynamical systems, representations, the important systems properties of controllability, observability, and identifiability. Section 4 addresses metamodeling issues such as the requirements for metamodeling simulations, limitations of metamodeling, representing discrete event systems and the determining the existence of a true input-output relationship. Section 5 introduces a structured metamodeling procedure by presenting an overview of the revised method that segments metamodeling into a sequential process. The remainder of the paper presents the key steps of the structured metamodeling method: Problem Definition, Section 6; Selection of the Metamodel Set, Section 7; Selection of the Identification Methodology, Section 8; and Generate the Metamodel, Section 9. Section 10 summarizes the paper. Symbols and notation used throughout the paper are included in an Appendix.

2 Metamodels

A model is a structure that can be used for understanding the behavior of a system [7]. The model can be a physical structure such as a wind tunnel model used to determine the aerodynamics of an aircraft or it could be a conceptual model represented by interactions, a system of equations or a simulation.

A simulation can be defined an instantiation or realization of a model. In this case the simulation is different from the model. We will use a more abstract definition. As stated, a model is a method of expressing a theory. The expression of the model – its representation – distinguishes sets of models. Therefore, we consider a simulation to be a particular representation of a model and will not distinguish between them.

Assume that we have a model of a system that cannot be used directly. A solution may not exist, it may be too complicated for a closed-form solution, it may require too much time to numerically determine a particular solution, or it may be a high-fidelity simulation that provides much more detail than we are interested in. Efficient use of this model requires a "black-box" approximation of the causal time dependent behavior

of the model – a metamodel.

For our purposes, then, a metamodel is a mathematical approximation of the system relationships defined by another, more detailed model.

As an abstraction, a metamodel is a projection of the model onto a subspace defined by new constraints or regions of interest. Selection of the parameters used for the projection (the construction of a metamodel) involves: *a priori* knowledge; the data; a set of metamodel sets; and rules to determine the best model to realize the data.

There are two general metamodeling techniques: the "Direct" and "Inverse" methods.

2.1 Direct Metamodeling

First, a metamodel could be developed by applying basic principles to generate a more abstract (approximate) version of the original model. This would be an example of direct modeling. Direct modeling is characterized by a specification of the elements of the model. Complicated systems are modeled by "tearing" a system into its components, modeling these components in a process called "zooming," and then interconnecting these components to construct a "physical" realization of the system [5, 8, 9]. The level of abstraction is controlled by the detail of the specification. The model reveals the structure of the theory and allows the prediction of the response to exogenous inputs as a function of the state of the system. The solution of this modeling problem requires an understanding of the process being modeled and methods to express this understanding.

Metamodels developed using this technique are "stand alone" versions. The relationship between the real system, the original model and the metamodel is contained in the two mappings from the underlying system to each of the models. There is no guarantee that a usable correspondence will exist between the metamodel and the model [10, 11]. Traceability from the high-fidelity model to the more abstract, lower fidelity metamodel becomes a significant issue. Also, this technique still requires an *a priori* understanding of the structure of the elements and the interconnections between these elements at the specific level of fidelity selected. This, in fact, could be a difficult and risky task and lack of this knowledge is often the reason that a high fidelity simulation was used in the first place.

2.2 Metamodeling via Solution of an "Inverse Problem"

Inverse modeling begins with the input-output data generated by the high fidelity model or simulation and develops the metamodel from the data. In this case, we have some estimate (measure) of the input and output response but do not have a complete characterization of the process by which the outputs are generated. While a properly posed direct problem generally has a solution, the inverse problem usually has multiple solutions out of which an acceptable solution (if it exists) must be selected. This technique explicitly results in a mathematical approximation between the inputs and responses.

It should be noted that there is a significant difference between our approach and much of the prior research. Most of the previous work that could be categorized as metamodeling consisted of procedures to determine the best polynomial fit to a set of input-output data. The researchers concentrated on the statistical properties of the data. In our approach, we are not trying to "fit" data. We are attempting to identify the underlying processes that define the system that generated the data (or in our terminology - the behavior). Therefore, the focus is not on statistics but on the system theoretic properties of the manifest behavior. In other words we are trying to identify the dynamical system that generated the data.

Dynamical systems acquire their importance from the fact that they exhibit memory or the potential to model phenomena where the past influences the future. A dynamical system is a family of trajectories without reference to I/O maps, variables, or behavioral equations. The system is coupled to its environment and is not defined by any associated model.

The metamodel is defined by the behavior it allows. This behavior is represented by inequalities or equations which can be grouped into sets. As we shall see in Section 7, selection of the proper metamodel set is critical to generation of an acceptable solution. There are several system properties that must be considered in the selection of the model set. These are controllability, observability and identifiability.

3 Identification Framework

Given a phenomenon that we would like to describe, we desire a mathematical expression as the model [5] [2]. Assume that this phenomenon produces outcomes that are elements of a set U. A model for this phenomenon will probably generate certain of these outcomes and exclude others. Consequently the outcomes recognized by the model B, are a subset of the universal set U, and are called the behavior of the model. For the inverse modeling problem, we define a model

[2] This framework follows the work presented by Willems.

class M with elements (U, B) where $B \subseteq U$ is the behavior of M.

Therefore, define a mathematical model as the pair (U, B) with U the universe of outcomes produced by the underlying phenomenon and B, the behavior of the model. If possible, we can describe the behavior of the model by a set of equations that leads to a behavioral equation representation of the pair (U, B). To accommodate this consider an abstract set E, called the equating space, and $f_1, f_2 : U \rightarrow E$. With this space, and the functions f_1, f_2, the behavioral representation for the model becomes (U, E, f_1, f_2).

In summary, the modeling procedure requires that we specify the variables in the phenomenon that we want to model (specify the universal set U) and identify the possible outcomes in the behavior, B. Often, however, we will require additional variables in addition to those we seek to model. These other variables are called latent variables. These variables are required whenever we develop a metamodel by the method of tearing where the system is viewed as the interconnection of subsystems. Consequently we expand the mathematical model to allow latent variables by defining a triple (U, L, B_l). Here L is the set of latent variables, $B_l \subseteq U \times L$, with $B_l \equiv \{u \in U | \exists l \in L : (u, l) \in B_l\}$. (Note: B_l could also be represented in an equating space as shown above.)

A mathematical model is linear if U is a vector space and B is a linear subspace of U. Assume that $U = I \times O$, where I is the input space, O is the output space, and B is the graph of a system map $S : I \times O$ called an I/O map. These assumptions allow an input-output model where $(U, B) \Leftrightarrow (I \times O, B) \Leftrightarrow (I, O, S)$. If the past does not contain any information about the future other than the information in the behavioral relationships, the map is nonanticipating. A parameterization of M consists of a set P and a surjective map $\pi : P \rightarrow M$. The set P is the parameter set with $p \in P$ determining the behavioral equations.

3.1 Dynamical Systems

Again, the model for a dynamical system is defined in terms of its behavior. A dynamical system is a family of trajectories without reference to I/O maps, variables or behavioral equations. The system is coupled to its environment and is not defined by a model associated with it. A model for a dynamical system Σ is simply a triple $\Sigma = (T, W, B)$ with $T \subseteq R$ the time axis, W the signal space, and $B \subseteq W^T$ the behavior – the set of all maps from T to W – a family of W-valued time trajectories.

A dynamical system is linear if W is a vector space (over a field F) and B is a linear subspace of W^T. A dynamical system $\Sigma = (T, W, B)$ is said to be time invariant if $\sigma^t B = B$ for all $t \in T$. Where σ^t is the time-shift operator: $(\sigma^t f)(t') \doteq f(t' + t)$.

A dynamical system $\Sigma = (T, W, B)$ is said to be complete if $\{w \in B\} \Leftrightarrow \{w|_{[t_1,t_2]} \in B_{[t_1,t_2]}, \forall t_1, t_2 \in T, t_1 \leq t_2\}$. Completeness is an important property affecting the mathematical structure that defines the behavioral equations that represent dynamical systems.

Dynamical systems acquire their importance from the fact that they exhibit memory or the potential to model phenomena where the past influences the future. In this context, a dynamical system is said to have a finite memory span $\Delta (\Delta \in T, \Delta > 0)$ if for $w_1, w_2 \in B, w_1(t) = w_2(t)$ for $0 \leq t \leq \Delta \Rightarrow \{w_1 \wedge w_2 \in B\}$ [3]. Where:

$$(w_1 \wedge w_2)(t) = \begin{cases} w_1(t) & for \quad t < 0 \\ w_2(t) & for \quad t \geq 0 \end{cases} \tag{1}$$

If $\Delta = 0$, the dynamical system is memoryless; if $\Delta = 1$ (in discrete time) the system is Markovian. Therefore, for a system with a finite memory span, the past is independent of the future. Σ is Δ complete $(\Delta \in T, \Delta > 0)$ if $\{w \in B\} \Leftrightarrow \{(\sigma^t w)|_{[0,\Delta]} \in B|_{[0,t]} \forall t \in T\}$.

Dynamical systems with latent variables and input/output dynamical systems can be defined in an analogous fashion as before. One method of representing latent variables is through state variables. A state-space dynamical system is defined as a dynamical system with latent variables, $\Sigma_l = (T, W, X, B_s)$ with $X \subseteq L$, such that the full behavior $B_s \in W \times X$ satisfies the axiom of state. In this case the latent variables, the states, contain sufficient information about the past so as to determine future autonomous behavior. The behavioral equations such as difference or differential equations, lead to representations of dynamical systems.

3.2 Representations

The model of the dynamical system is defined by the behavior that it allows. The behavior can be defined by a set of inequalities or equations. The structure of the equations is a representation of the model.

A representation that is only a function of current and past signals (outputs) and is called an autoregressive (AR) representation and can be written as $R(\sigma^L, \sigma^{-l})w = 0$. Where

$$R(\sigma^L, \sigma^{-l}) = R_L s^L + R_{L-1} s^{L-1} + \ldots + R_{l+1} s^{l+1} + R_l s^l \tag{2}$$

[3] Here $\wedge$ denotes concatenation

If the system that we are trying to model suggests latent variables to describe the behavior, the autoregressive representation can be expanded to include a moving average part of the past latent variables resulting in an autoregressive-moving-average (ARMA) representation. In this case, the behavioral difference equations relate the time-series of the manifest variables $w : Z \rightarrow R^q$ to the time-series of the latent variables $x : Z \rightarrow R^q$. With appropriate definitions, the ARMA system is defined as:

$$R(\sigma^L, \sigma^{-l})w = M(\sigma^L, \sigma^{-l})x \qquad (3)$$

An important class of ARMA systems are those where $R(s, s^{-1}) = I$. This yields a moving average (MA) representation: $w = M(\sigma, \sigma^{-1})x$

We can combine the above constructs to define a class of models with all of the advantages of completeness – described by the difference equation; state form – the memory is displayed through the latent variables; and nonanticipating input-output – an explicit cause and effect structure. This representation is an input/state/output representation and is the model class most amenable to analysis, synthesis and simulation.

3.3 Controllability, Observability, and Identifiability

In a controllable system, the past trajectory does not have a lasting influence on the far future. Sooner or later any other trajectory within the controllable subspace can be attained [12]. All dynamical systems are not controllable. In an autonomous system, the past trajectory determines its future completely. Consequently, the lack of controllability implies predictability. As we develop the capability to better understand and control our environments, our ability to predict that environment can suffer. We are limited in our ability to predict by our ability to observe.

Observability is the ability to reconstruct the trajectory of latent variables from the manifest set [13]. While controllability is intrinsic to the dynamic system, observability is also a function of the representation of that system. This comes about because observability is only an issue for dynamical system model representations that have latent variables (by definition if the variable is a manifest variable it is observed) and is a property where an unobserved signal can be deduced from one which is observed.

Identifiability relates to the ability to reconstruct the dynamical laws of the system from a given set of measurements [14]. There are several obstructions to identifiability. Feedback makes it difficult to separate system dynamics from the dynamics of feedback. Structured inputs can interfere with the structure of the behavior. The failure of the input to excite all of the modes will prevent observation (and subsequent identification) of the unexcited modes. And finally, over parameterization can result in dependencies that cause gradients to become singular preventing identification of the system.

Any persistently exciting unstructured input will be sufficiently rich to observe a controllable system. Structured inputs will allow observation and identification if the AR relations defining the structure of the input have large lags that do not interfere with the structure of the system. In other words: if the structure of the input is not seen by the system [5].

In order to identify a portion of a system, we must be able to observe the response. Observability specifies the ability to determine the trajectory of latent variables from the manifest set. Since controllability allows an MA representation, and any controllable MA representation can be converted into an AR representation by increasing the lag, complete controllability implies observability. Lack of controllability, however, does not imply lack of observability [15]. For systems that can be reduced to an AR-representation, $R_1(\sigma^L, \sigma^{-l})w_1 + R_2(\sigma^L, \sigma^{-l})w_2 = 0$ with $R_1[\sigma^L, \sigma^{-l}] \in R^{g \times q_1}[\sigma^L, \sigma^{-l}]$ and $R_2[\sigma^L, \sigma^{-l}] \in R^{g \times q_2}[\sigma^L, \sigma^{-l}]$ then w_2 is observable from w_1 if and only if the rank of the matrix $R_2(\sigma^L, \sigma^{-l})$ is equal to $q_2 \forall \sigma \neq 0$.

This is why inverse modeling or system identification is thought to be difficult – the system and our selection of a representation is critical in that it constrains the behaviors of the model, affects our ability to observe latent variables, impacts our ability to represent the outcomes U and defines our ability to identify the underlying processes.

3.4 Discrete-Event Systems (DES)

The above framework is consistent with the formalized discrete-event systems in theoretical computer science. The behavior is similar to the formal language; a state-space system is like an automation; latent variables are replaced by production rules; interconnections are communications. The most significant difference is the lack of behavioral models (equations) in the theory of DES. Also completeness is usually violated in a DES by initiation and termination rules for event strings.

Since the DES is not complete, representation of these systems requires special consideration. We will see in Section 4 that completeness is required to represent a dynamical system by a behavioral difference equation. Results for representation of complete sys-

tems may be generalized to a class of noncomplete systems (including DES) that meet specific restrictions.

The linear time-invariant dynamical system, (Z, R^q, B), is called an l_2-system if B is a linear shift-invariant closed subspace of $l_2(Z; R^q)$. Define $\overline{B^{pc}}$ as the closure of B with respect to the topology of pointwise convergence. This corresponds to finite dimensionality. With these definitions, results for complete systems may be generalized to l_2-systems satisfying $B = \overline{B^{pc}} \bigcap l_2(Z; R^q)$[4].

4 Metamodeling Issues

With a framework established to characterize system models, we now address the key issue of the inverse modeling problem: "What properties of the behavior allow the system to be represented by a difference (or differential) equation of a particular type?" Analysis of these properties will result in rules and constraints for the setup and design of metamodels. Since we are no longer fitting data but identifying systems, the data used to identify the system must meet certain prescriptions. Explanation and proof of the following statements can be found in [5].

1. To represent a system by means of a difference equation it has to be complete (it cannot have initialization or termination conditions at $t = \pm\infty$) with a finite memory span so that observation of a trajectory on a finite time interval allows conclusions about past behavior independent of what will happen in the future.

2. For a system to be described by AR-equations it must be linear, complete and time invariant.

3. Since a dynamical system containing latent variables can be converted into an AR representation with an increase in the lag, representation of a dynamical system with latent variables must also be linear, complete and time invariant.

4. If the dynamical system is controllable (if it is possible to eventually steer the system to a desired trajectory) then the system will also allow an MA representation.

5. An input-output dynamical representation can be defined if, and only if, it can be described by an AR-system of behavioral equations $P(\sigma, \sigma^{-1})y = Q(\sigma, \sigma^{-1})u$ with $P(s, s^{-1}) \in R^{p\times q}[s, s^{-1}]$, $Q(s, s^{-1}) \in R^{p\times m}[s, s^{-1}]$ and $\det P \neq 0$.

6. The input-output dynamical representation will be nonanticipating if and only if $P^{-1}(s, s^{-1})Q(s, s^{-1}) \in R^{p\times m}(s)$ is a matrix of proper rational functions.

7. The manifest behavior of the state variable (an ARMA) representation will belong to L^q. Consequently every system $\Sigma \in L^q$ admits a finite-dimensional state representation, allows a componentwise I/O representation and admits an Input/State/Output representation.

4.1 Metamodeling Simulations

In this paper we consider inverse modeling and concentrate on the metamodel sets and rules to determine the best model.

With respect to metamodeling simulations, the systems we are trying to identify are complex, nonlinear and time-varying. They can be continuous, discrete, sampled-data (continuous systems with discrete measurements) or discrete event systems. In general, for these cases, the predictor function is a nonlinear function of past observations and there are too many possibilities for unstructured "black box" models. Knowledge of the nonlinearities must be built into the model [16].

Care must be taken in the setup of the Metamodeling problem. The experimental design must provide input-output sequences that correctly represent the system structure. When the metamodel is determined, it is not possible to ask "What is the probability that a particular set of fitted parameters is correct?" because there is no statistical universe of models from which the correct one is chosen. There is just one model and a statistical universe of data sets that are drawn from it. It is possible, however, to ask "Given a particular set of parameters, what is the probability that this data set could have occurred?" We can identify the probability of the data given the parameters as the likelihood of the parameters given the data [17].

Fortunately, in our case we have explicit knowledge of the nature and characteristics of the high fidelity system. We have the model (the simulation) that applied the system to the inputs to generate the outputs we are interested in. Given this information we can build the nonlinearities into the structure of the metamodel and provide the capability to generate a reduced order approximation of the original model. This fact makes metamodeling as a method of model abstraction feasible. We exploit this fact to the fullest extent possible.

In addition to knowledge of the nonlinearities, other requirements must be met to allow representation of

[4] See [10] and [11] for definitions of l_p spaces.

the system by a finite dimensional, reduced order approximation: the system must be complete; the axiom of state must apply; and the output must be nonanticipating.

Simulations are usually defined to represent real-world events that have a beginning and an end. Given that the simulation terminates naturally, results for complete systems can be applied since the system behavior is restricted to a finite dimensional sequence.

In general the axiom of state applies because the simulation is set up in such a way that the initial conditions contain sufficient information about the past so as to determine future autonomous behavior.

Also the presence of input and output files indicates that an input-output structure with causality is assumed in the model represented by the simulation.

In summary, assuming that the underlying system modeled by the simulation is well behaved, (Markovian, complete with respect to the modeled behavior) the following is required to metamodel simulations:

1. The data must include the behavior we are trying to model [18].
2. The latent variables that define the behavior must be observable [16].
3. The input must be persistently exciting so that the effects of the latent variables are observed [19].
4. For a stochastic system, the ensemble of trajectories must span the space [20].
5. Any single trajectory must span both the input and output space and be sufficiently long so that the state transition probabilities also span the allowable probability space and the distribution of these probabilities are the same as the underlying system [16].

4.2 Metamodeling Limitations

In addition to the problem setup and experimental design, the metamodel solution comes with limits of its own. Using the space spanned by the original model as the full order model, the metamodel is a reduced order approximation. This reduction inherently limits the span of the manifest (exogenous) variables associated with the behavior (input or output - if such a map exists). Consequently, the behaviors allowed by the metamodel will exist within a subspace of the original model.

Assuming that an input-output map exists for the model, input values will be restricted to a domain within which the metamodel will be applicable. Outside of this hypersurface, application of the metamodel may provide numbers but will not generate an output that is representative of the modeled system. Also, assuming appropriate inputs, the output of the metamodel can only be guaranteed to be approximately correct. As a projection, the metamodel will not contain all of the detail of the original model. There are output error bounds that are a function of both the metamodel and the input.

4.3 Representing Discrete Event Systems

The discussion above introduced the issues associated with Discrete Event Systems. Most of system identification is formulated on continuous, discrete or continuous-discrete dynamical systems. Many of the simulations are discrete event or connected discrete-event dynamical systems. The question arises: "When can a DES be described by a difference equation?"

Since completeness is usually violated this impact must be expressly considered. If a linear time-invariant system is not complete then whether or not $w : Z \rightarrow R^q$ belongs to the behavior depends on $w(t)$ at $\pm\infty$. However, results for complete systems can be generalized if the system behavior is restricted to a finite dimensional sequence. From Section 3.4, every behavior $B \in L^q$ allows an AR representation. Define a DES as a time-invariant system $\Sigma = (Z, W, B)$ with W a finite set. A DES is internally finite if it can be realized by a finite automation and/or if there exists a state-space representation of it with a finite-state space. An internally finite and complete DES $\Sigma = (Z, W, B)$ can be described by a behavioral difference equation $f \circ (\sigma^L w, \sigma^{L-1} w, \ldots, \sigma^l w, w) = 0$ for some $L \in Z$ and some $f : W^{L+1} \rightarrow \{0, 1\}$.

4.4 Existence of a true Input-Output Relationship

Assume that we have observed the input and output of a system and computed a set of linear differential and/or algebraic equations from this data. Have we identified the system? Do these equations establish a true input-output relationship suggested by this identification? For linear systems, answers to these questions are provided by two sequences of subspaces, one in the input space u and the other in the output space y [21].

Consider a system of linear ordinary differential and algebraic equations with constant coefficients: $A(\sigma)\xi + B(\sigma)u + C(\sigma)y = 0$ where (σ) denotes differentiation (or the shift operator for discrete time systems) and ξ contains all of the latent variables not present in the input and output spaces. $A(s)$, $B(s)$, and $C(s)$ are polynomial matrices.

We say that y processes u if the linear space of trajectories $\{y|(y,0) \in B\}$ is finite dimensional. Therefore, y processes u if u determines y up to a finite number of constants. Also u is free if for every trajectory u there exists a trajectory y such that $(y,u) \in B$.

Recall that if the dynamical system with latent variables $\Sigma_l = (Z, R^q, R^d, B_l)$ is linear time invariant and complete, the manifest system which it represents $\Sigma = (Z, R^q, B)$ is also linear time invariant and complete. Consequently, for a linear time invariant and complete system, any behavior given by $A(\sigma)\xi + B(\sigma)u + C(\sigma)y = 0$ can also be represented by:

$$B = \left\{ \left| \begin{bmatrix} y \\ u \end{bmatrix} \right| [R_1(\sigma) \;\; R_2(\sigma)] \begin{bmatrix} y \\ u \end{bmatrix} = 0 \right\} \quad (4)$$

The behavior of such a set of equations stems from an input-output system if both conditions of the following proposition hold.

Proposition 1 *Let a behavior B be given by equation 4. where $[R_1(\sigma) \;\; R_2(\sigma)]$ is a polynomial matrix of full row rank. The following statements hold:*

1. y processes u if and only if $R_1(s)$ has full column rank.
2. u is free if and only if, $R_1(s)$ has full row rank.

Therefore, $R_1(s)$ must be invertable and the transfer matrix of the system is defined by $T(s) = -R_1^{-1}(s)R_2(s)$.

Consequently once the identification is accomplished, the subspaces generated by the system (equation 4) can be checked to determine if a true input-output relationship has been found (Refer to [21] for additional detail).

5 General Approach

Thus far we have presented the new approach to addressing metamodeling problems and a framework for applying that approach. We now discuss a structured method for implementing the framework that is based on a new taxonomy of metamodel sets and identification methods.

The development of this method began with the following metamodeling procedure presented in [1]:

1. Determine the purpose of the metamodel.
2. Identify the response.
3. Identify important response characteristics.
4. Identify input factors.
5. Identify important input characteristics.
6. Specify the experimental region.
7. Select validity measures.
8. Specify required validity.
9. Postulate a metamodel based on:
   Input - Output response characteristics.
   Experimental region dimensions.
   Required validity.
10. Select an experimental design.
11. Obtain data.
12. Fit the metamodel.
13. Assess the validity of the model.

This procedure was primarily based on the use of least squares to realize the metamodel. When applied to our new framework this procedure, especially "Step 9: Postulate a metamodel," resulted in too many complex decisions involving: *a priori* knowledge; the data; possible metamodel sets; and rules to determine the best model set to realize the data.

In order to categorize metamodeling problems and their solutions in a manner that would allow the separation of the metamodeling process into a set of sequential decisions based on *a priori* information each of these areas were analyzed to derive a taxonomy that would support a structured metamodeling procedure.

The first eight steps of the metamodeling procedure provide the prior knowledge or metamodel requirements that define the problem. The remaining steps define the experimental setup, the model set, the method of identification and validity measures used to develop and verify the metamodel. This fact was used to separate the procedure into two general areas. The first eight became the foundation for the problem definition; the remaining steps were grouped as iterative steps in the metamodeling process shown in Figure 1.

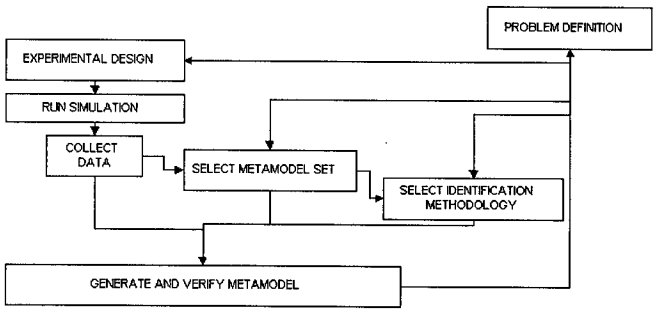


Figure 1: Iterative Metamodeling Process

In this process "Step 9: Postulate a metamodel" was decomposed into the two steps "Select a Metamodel Set" and "Select Identification Methodology." Both of these steps are a function of both the problem definition and the data generated by the simulation.

"Step 10" was directly incorporated in the structured method. Procedures for defining the experimental design, for pretreatment of the data and for the verification processes are covered in references such as [16, 22, 23] and are further discussed in [24].

"Step 11: Obtain Data" is addressed by the two steps in the revised process: "Run the Simulation" and "Collect Data." These steps are purely mechanical in nature and are not discussed further.

We now present the key steps of the structured metamodeling method: "Problem Definition," Section 6; "Selection of the Metamodel Set," Section 7; "Selection of the Identification Methodology," Section 8; and "Generate the Metamodel," Section 9.

6 Problem Definition

We define a metamodeling problem as the direct sum of the metamodel requirements and the model (simulation). This means that the same simulation could be part of two different metamodeling problems if the requirements were different. Conversely the same set of requirements applied to two different (nonsimilar) simulations also leads to two different metamodeling problems.

Consequently, to define the problem we must consider both elements of the direct sum – the purpose of the metamodel and the simulation characteristics.

6.1 Metamodel Purpose

As mathematical relationships metamodels can be developed to support two general purposes: (1) Analysis; or (2) Hierarchical simulation.

First, a metamodel can be used for analysis. In this case the metamodel becomes an independent structure that is used to understand and extract information from the model. Table 1 depicts the scope and uses of analytical metamodels.

Table 1: Scope and Uses of Analytical Metamodels.

Scope	Approximate an Unknown Response Model a Simulated Process
Uses	Sensitivity Analysis Estimation of Existing States Predict and Control Future Responses Optimize Expected Performance Diagnosis of Faults

Table 2: Scope and Uses of Simulation Metamodels.

Scope	Develop Atomic Simulation Components Build Coupled Simulation Components
Uses	Execution Speed Maintainability / Configuration Control Verification, Validation, and Accreditation

Secondly, a metamodel can be used to support hierarchical simulation and model reuse (Table 2). In this case the metamodel is used in conjunction with (coupled to) other simulations or simulation elements to answer larger questions that are not supported within the structure of the modeled simulation. This purpose supports simulation based on a hierarchical representation. Using metamodels for this purpose is a two-step process. First a metamodel of a simulation (or component) is generated to develop more abstract simulation model. Once developed, interface modules can be used to couple these metamodels to other simulations or metamodels to represent a more complex system.

The selection of scope and use defines the metamodel purpose and provides clear boundary conditions for follow-on selections in Steps 2 through 8 which are also part of Problem Definition.

6.2 Simulation Characteristics

We have discussed the purpose of the metamodel. Since all of the remaining problem definition decisions are a function (direct sum) of both the metamodel requirements and the simulation that is to be modeled, we concentrate on the aggregate space of simulation characteristics. Research has suggested that both a general (external) description of the simulation or model as well as further detail on the (internal) process structure of the internal components is required (Refer to [3, 4]).

The classification defined by the "SIMTAX, A Taxonomy for Warfare Simulation" was completely adequate for the external description. This taxonomy was developed by the Military Operational Research Society (MORS) and addresses three equally important relational dimensions: the purpose, the qualities and the construction of the model or simulation [25]. It is a descriptive framework designed to guide the development, acquisition and use of warfare models and provides the basis for classifying objects for identification, retrieval and research purposes.

Selection of a metamodel set requires detailed information not contained in the simulation and model catalogues. Recall that our new approach and framework concentrates on the behavior of the underlying system that defined the process. The metamodel representation of the simulation is realized by the parameterization of the selected metamodel set. The performance of the metamodel is directly correlated to the match between the behavior of the underlying system and the metamodel set. To provide a link between the simulation behavior as described by the more general taxonomy outlined above and specific metamodel sets a more detailed internal taxonomy was appended to the SIMTAX. The purpose of this additional detail is to link the behavior (of the simulation) and the representation (of the metamodel) and uses system theoretic definitions common to control engineering.

Figure 2 depicts the model of a continuous system with a sampled measurement. In development of a metamodel we try to isolate and identify each of the individual elements in this model. Consequently we must be able to characterize the type of processing that takes place in each of the blocks.

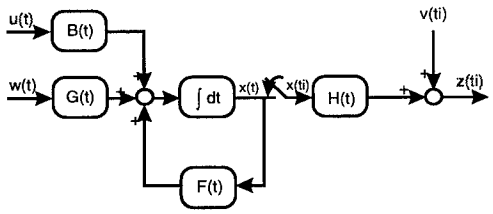


Figure 2: System Model.

To explain this concept, we will consider the model of an aircraft. The input, $u(t)$, is the pilot or autopilot command. In modern aircraft this would be a desired acceleration ("g") or angle of attack ("α"). In older aircraft it would be something closer to the flight control surface such as the torque necessary to hold the control surface in a given position. This input is acted on by B(t) to provide the input expected by the plant. In an inertial frame it could be the force applied. In a more complicated model it could be the control surface deflection. Another input path accepts input disturbances w(t) (e.g. turbulence). The plant, represented by $F(t)$, is the model of the physical system. In the case of the aircraft, it could be something as simple as $F = ma$ (if the simulation was completely defined in an inertial frame) or it could be the body axis stability derivatives that make up the coefficients in the equations of motion [26]. The noise corrupted output, $z(ti)$, is the measurement available. The instrumentation system that performs these measurements is represented by $H(t)$. The aircraft would have accelerometers or an inertial system that measures the body axis accelerations or inertial position and attitude. The combination of all of these blocks represents a single entity in a simulation.

Formulating the metamodeling problem with this additional detail is important for two reasons. First, each of these blocks may be represented by a separable process and it is usually not possible to simultaneously identify more than one process.

If we try to simultaneously identify two processes and the processes are independent, a rank deficiency in the uncoupled equations causes numerical difficulties. If the processes are dependent, behaviors associated with both processes will be combined preventing the identification of either.

If one is successful in simultaneously identifying multiple process, performance of the resulting metamodel is usually poor. Unless the model set and order accurately accommodates both systems, the minimization process used for identification will generate a system of equations that represents the combined behaviors within allowable tolerances but represents neither system well.

Returning to the internal taxonomy, categories and selections for these categories that were used to provide the additional detail on the internal structure are shown in Table 3 and are described below.

Basis. This is the fundamental basis of the simulation. The simulation will model either a physical phenomenon or will model events that simulate human or system interactions with its environment. Simulations that are a combination of the two, default to event based (the more complex of the two basis).

System, Input, and Output Processing. This concerns the plant or system that is modeled. It covers the algebraic structure (Linear or Nonlinear) and realizations (Stochastic or Deterministic) used to process the inputs, describe the plant and the method of generating the observed output. Each of these elements is considered independently.

Result/Trajectory. The output of the simulation can be a single trajectory that maps a series of events or it can be a result that is incorporated into a statistical database to determine probabilities of occurrences.

Level. This describes the class of the system and the types of representations that can be used. SISO is single-input single-output system. A MISO system has multiple inputs but still a single output. A MIMO system is the most complex with multiple inputs and multiple outputs.

Process description. In a "Complex" simulation there are inputs to more than one separable process

Table 3: Internal Processing Description.

Basis	Physics based Event based
System Inputs Outputs	Linear Nonlinear Stochastic Deterministic
Result/Trajectory	Functional Statistical base
Level	SISO MISO MIMO
Process description	Complex Simple Coupled
Interval	Continuous time Discrete time Continuous - discrete time Discrete-event

(system). In a "Simple" simulation there are inputs to only one process (system). There is no additional influence on the system (other than predefined parameters). In a "Coupled" simulation there are inputs to only one process (system) but there are additional non-deterministic impacts on the output.

Interval. There are three ways that a dynamical system can evolve in time. It may be continuous (analog), discrete or it can be discrete-event. Also there are two options for measuring (sampling) the output of a continuous system. It may be sampled continuously or discretely (at specific time intervals).

6.3 Problem Definition Summary

At this point, we have determined the purpose of the metamodel. In the definition of this purpose we have identified the input and response that we are interested in and determined the important characteristics of these data. Also for this purpose, we have defined the region of interest, selected validity measures and specified the required validity.

In addition we have characterized the simulation that were are trying to model. In keeping with our new approach, we have not addressed the representation of the metamodel or assumptions that will be made in it's realization. However, data generated by this step provides a clear statement of the metamodel purpose and the characteristics of the simulation that will be modeled. As will be seen in the next section, this data directly matches the decisions that must be made in the selection of the model set.

7 Selection of the Metamodel Set

Now we discuss decisions associated with "Select a Metamodel Set." The completion of this step requires a number of interrelated selections. So many options are available that the combination of model selection, error criterion, identification technique, and numerical methods leads to an overwhelming myriad of "identification methods."

In fact there seem to be as many system identification methods as there are inverse problems. Many specific identification and statistical methods have been developed to accommodate the differences in model structures, data length, measurement error statistics, etc. Also, the literature contains considerable discussion on particular methods with very little discussion on the relationship of these techniques to each other or to a general methodology. The result is a confusing array of unconnected methods with little or no guidance on the application of the techniques to general classes of problems.

Since we are looking for procedures to handle general metamodeling problems, we discuss these methods as elements of a more general structure and have reduced these selections to four (including Selection of the Identification Methodology - Section 8) that best match the characterization of the simulation to the behavior allowed by the metamodel set.

In reality all "real world" systems are complex, large scale interconnections of continuous-discrete, nonlinear, infinite-dimensional components. We will approximate these systems with lumped parameter, parametric, finite dimensional models that can be grouped into sets.

A metamodel set is defined by the system description, system class and metamodel structure (representation). For any given problem, multiple model sets are available. In each of these model sets a most powerful unfalsified model [5] will exist (given that the requirements of Section 4 are met) [5]. Consequently the performance of the metamodel will be limited by the match between the metamodel set and actual system that generated the behavior.

7.1 System Description

In the definition of the system description, the first selection concerns the system type that will define the allowed behavior of the models. Here the most basic

[5] Defined in Section 9.

Table 4: System Description.

Selection	Options
Type	Static Dynamic - Time Invariant Dynamic - Time Varying
Algebraic Structure	Linear Nonlinear
Randomness	Stochastic Deterministic
Time	Continuous time Discrete time Continuous - discrete time Discrete-event

Table 5: System Classes and Representations.

MODEL CLASS	FORMS OF THE REPRESENTATION
SISO MISO MIMO	Polynomial Matrix Fraction State Space

questions must be addressed. How are the parameters described? Is the representation going to include dynamics or will it be static? Will the model contain latent variables? If it is dynamic, is it time invariant or time varying?

Is the algebraic structure linear or nonlinear? Are disturbances, noise and randomness accommodated? Is the system defined as continuous, discrete, continuous-discrete or as a discrete-event system? Table 4 outlines the possible selections that define the system description. Note that while both static and dynamical models can both accommodate nonlinear and stochastic behavior, only dynamical systems have time and trajectories associated with them.

7.2 System Class

In addition to the system description, the class of the representation is also needed to define the overall model set. This class is defined by the interaction of the variables and the representation. Table 5 provides a list of the general system classes and the possible form of the representations [19, 20].

Comparing Tables 4 and 5 with Table 3 we see that the characteristics of the behavior we are modeling define the first two elements of the metamodel set: the system description and the system class.

7.3 Metamodel Structure

Once a system description and class that match the underlying behavior have been selected, the next decision is selection of the model structure to use in describing the response of the system to the inputs (possibly including latent variables). A metamodel structure $\mathcal{M}$ is defined as a differentiable mapping from a connected open subset D_m of R^d to a metamodel set $M(\theta)$, such that the gradients of the predictor functions are stable. In this definition θ is the vector used to parameterize the model and D_m is the values over which θ may range in the metamodel set $M(\theta)$. There are many metamodel structures available and this area generates much of the complexity in system identification.

We simplify this decision by defining two general model structures; predictor models and probabilistic models. A predictor model only defines the predictor equation(s). These models specify the elements of the transfer function in terms of some parameter set. The models generated from these structures are deterministic in nature. [6]

A probabilistic model accommodates the fact that many systems are subject to known disturbances that are not (or cannot be) completely categorized. The statistics of the noise and disturbances are included as random variables. Probabilistic models supplement the parametric description with a description of the density function (or moments) of the noise (disturbance) that acts on the system. The variables of the system being identified become functions of random variables. In these situations different realizations of an experiment (simulation run) may not produce exactly the same results. Consequently the output of a probabilistic model is the conditional expected value and the joint or conditional probability density functions (JPDF or CPDF) of the variables.

The following two subsections discuss these two model structures. Since all models are not appropriate for every system description and class the available selections are dependent on the description and class we have selected.

7.3.1 Predictor Models

The selection of the metamodel structure should match the system description. Since predictor models are used for deterministic systems we provide models that match the type and algebraic structure shown in Table 4.

Static. Static systems can be either linear or nonlinear. The predictor equations for static models are

[6] Predictor models do allow for the prediction or measurement error. And since the coefficients were generated via a minimization of some error criterion with assumed statistics, the coefficients will be random variables with an error distribution. Since the estimates are functions of these random variables, this distribution can be used to compute error bounds of the estimate.

the actual input-output map that comes from the selected representation and are similar to those representing dynamical systems. Also static models can be set up using dynamical model structures with a zero state transition.

Dynamic. For dynamic systems, we use the model structure to predict the output of the model. The differences between this prediction and the actual data are then used to arrive at the parameter set which minimizes the error. As the complexity of the system description increases, the flexibility in the selection of the representation (polynomial, matrix fraction description, state space) decreases.

We will consider three types of dynamic models: linear time-invariant, linear time-varying and nonlinear. To save space, discrete realizations are presented. However, continuous realizations can also be used. All nonlinear systems will be assumed to be Markov.

Linear Time-Invariant Predictor Models. There are a number of ways of defining the transfer function (input-output map) associated with linear time-invariant predictor models. First, the numerator and denominator polynomials of the transfer function can be given explicitly in either discrete or continuous time. This polynomial transfer function can also be converted into a frequency function that gives the frequency response of the input-output map. Finally, the transfer function can also be defined by the zeros and poles of the model. These descriptions are most appropriate for SISO systems. [27]

MISO systems are best represented by a state space or polynomial format that explicitly defines the coefficients of each of the input and output terms.

Our general linear metamodel structure is:

$$y(t_i) = G(q)u(t_i) + H(q)e(t_i) \tag{5}$$

where $y(t_i)$ is the output, $u(t_i)$ is the input, and $e(t_i)$ is the error. $G(q)$ is the transfer function between the input and output, while $H(q)$ is the transfer function between the error term and the output. Here q^{-1} is the backward shift operator so that $q^{-1}u(t_i) = u(t_{i-1})$. As a result the polynomials have the form $G(q) = 1 + g_1 q^{-1} + \cdots + g_{n_g} q^{-n_g}$.

From this general model, we can define a SISO or MISO model structure as:

$$A(q)y(t_i) = \frac{B(q)}{F(q)}u(t_i) + \frac{C(q)}{D(q)}e(t_i) \tag{6}$$

The predictor for this general polynomial structure is:

$$\hat{y}(t_i|\theta) = \frac{D(q)B(q)}{C(q)F(q)}u(t_i) + \left[1 - \frac{D(q)A(q)}{C(q)}\right]y(t_i) \tag{7}$$

where each of the polynomials are a function of the parameter set θ. Latent variables (that are not past values of the input or output) can also be defined in the polynomial format by augmenting the input-output relationships to include the additional variables.

Now consider the situation where the input is a m-dimensional vector, and the output is a p-dimensional vector – a multiple-input-multiple-output (MIMO) system. In this case, the term $\frac{B(q)}{F(q)}$ has no meaning. While a matrix fraction description (MFD) or state space representation can be used, MIMO systems are most amenable to the state space format. (See [14, 28] for a discussion of MFDs) This format also has the most flexibility in defining the relationship to latent variables. In this (discrete) description we add the state variable $x(t_i)$ that is propagated forward in time by:

$$\hat{x}(t_{i+1}|\theta) = A(\theta)x(t_i, \theta) + B(\theta)u(t_i) \tag{8}$$

and the measurement equation:

$$\hat{y}(t_i|\theta) = C(\theta)x(t_i, \theta) + D(\theta)u(t_i) \tag{9}$$

that provides the output.

One of the most flexible state space predictor models is the directly parameterized innovations form. Based on the classical steady state Kalman filter, this model accommodates the fact that measurement and process noise are present but does not require knowledge of the disturbance properties. This is accomplished by parameterizing and identifying the Kalman Gain instead of the process and noise descriptions: [7]

$$\hat{x}(t_{i+1}, \theta) = A(\theta)x(t_i, \theta) + B(\theta)u(t_i) + K(\theta)\left[e(t_i)\right] \tag{10}$$

Relating this to the general model structure given above we see that:

$$G(q, \theta) = C(\theta)\left[qI - A(\theta)\right]^{-1}B(\theta) \tag{11}$$

$$H(q, \theta) = C(\theta)\left[qI - A(\theta)\right]^{-1}K(\theta) + I \tag{12}$$

Linear time-varying models. Linear time-varying systems are restricted to weighting function and state space forms. Predictor metamodels for use with a weighting function have the same form as metamodels used for time-invariant systems except that the weighting function is time varying. Time-varying state space models are similar to the time-invariant state models with the exception of the time index on the coefficients [16].

Nonlinear Models. With respect to general dynamical nonlinear models the situation is far too flexible. The output may be a function of all of the past inputs and outputs yet we are going to represent this system with a finite number of parameters. Usually

[7] The error $e(t_i|\theta) = y(t_i) - C(\theta)\hat{x}(t_i, \theta)$.

considerable insight is required to effectively use a nonlinear model type.

Systems with linear dynamics and static input nonlinearities can be handled by redefining input of the system to exclude this nonlinearity (Hammerstein model). With this new definition the system can be identified by a linear model. Another powerful procedure for developing a nonlinear model (given the required insight into the problem) is to build the nonlinear model as a nonlinear combination of linear systems (e.g. use the output of a linear system as the input to another linear system).

Nonlinear systems (that are not approximated by linearization, perturbation, or combination) are restricted either to a pseudolinear form or state space descriptions. We define the pseudolinear form as $\hat{y}(t_i|\theta) = \theta^T \phi(t_i)$ where θ^T is the vector of unknown coefficients and $\phi(t_i)$ contains the nonlinear combinations (functions) of the input data. Although the structure looks static dynamics can be included in the pseudolinear model by including nonlinear combinations of past data.

If we want to explicitly consider system dynamics for nonlinear predictor models there is only one option: a nonlinear state space or a simulation model. (Note: Nonlinear state space and simulation models are not the same for Probabilistic models where disturbances are explicitly considered.) The nonlinear state space model is defined as:

$$\begin{aligned} x(t_{i+1}|\theta) &= f(t, x(t_i), u(t_i), \theta) \\ y(t_i|\theta) &= h(t, x(t_i), u(t_i), \theta) \end{aligned} \tag{13}$$

7.3.2 Probabilistic Models

Models for probabilistic descriptions will be limited to the state space form. While transfer function and matrix fraction descriptions are limited to linear time-invariant systems, a state space system does not share this restriction. This form also allows the combination of a continuous system with discrete measurements (a sampled-data system) to more closely match real systems.

We cover four types of probabilistic models. The first type of model is a linear stochastic model developed by assuming a white noise approximation. The second model is a general nonlinear stochastic model. The third type is a linear Ito stochastic model based on the correct description of the noise as Brownian motion with an Ito stochastic description and the final model is a full nonlinear Ito stochastic model.

Linear Stochastic. Linear stochastic system modeling results in the following model driven by known inputs and white noise $w(t)$ [29]:

$$\dot{x} = F(t)x(t) + G(t)u(t) + L(t)w(t) \tag{14}$$

starting from a Gaussian $x(t_0)$ with a known mean $\hat{x}_0$ and covariance P_0. Average performance can often be described by this simple stochastic differential equation sometimes referred to as Langevin's equation [30, 31].

This model is supported by a discrete (or possibly continuous) linear measurement corrupted by additive white noise $\nu(t_i)$:

$$z(t_i) = H(t_i)x(t_i) + \nu(t_i) \tag{15}$$

The noise processes were assumed independent of the initial condition and (at least initially) each other with zero mean and correlation kernels given by:

$$E\{w(t)w^T(t+\tau)\} = Q(t)\delta(\tau) \tag{16}$$

$$E\{w(t)\nu^T(t+\tau)\} = S(t)\delta(\tau) \tag{17}$$

$$E\{\nu(t_i)\nu^T(t_j)\} = R(t_i)\delta_{ij} \tag{18}$$

Since the solution of these systems is a stochastic process with many potential realizations, it is best to characterize the system by the expected value of its moments (mean, variance, etc.) The optimal (minimum mean square error, unbiased, consistent) predictor for this system is the classical Kalman-Bucy Filter [33].

Continuous Time Predictor. The continuous-time predictor consists of the following set of equations:

State estimate:

$$\dot{\hat{x}}(t) = F(t)\hat{x}(t) + G(t)u(t) + K(t)\left[z(t) - H(t)\hat{x}(t)\right] \tag{19}$$

Filter gain calculation:

$$K(t) = \left[P(t)H^T(t) + L(t)S(t)\right] R^{-1} \tag{20}$$

Error covariance propagation (Riccati equation):

$$\begin{aligned} \dot{P}(t) &= F(t)P(t) + P(t)F^T(t) + L(t)Q(t)L^T(t) \\ &\quad -K(t)R(t)K^T(t) \end{aligned} \tag{21}$$

Discrete Time Predictor. The discrete-time predictor includes an additional step beyond those required for the continuous filter [29, 32, 33]. Given a state and covariance estimate with discrete noise processes Q_d and R_d, those estimates are first extrapolated to the next time step (without taking a measurement). These estimates are represented by time (t_i^-). At the next time step a measurement is taken and the estimates are updated. These estimates are identified by the time (t_i^+).

State estimate extrapolation:

$$\hat{x}(t_i^-) = A(t_{i-1})\hat{x}(t_{i-1}^+) + B(t_{i-1})u(t_{i-1}) \tag{22}$$

Error covariance extrapolation:

$$P(t_i^-) = A(t_{i-1})P(t_{i-1}^+)A^T(t_{i-1}) + Q_d(t_{i-1}) \quad (23)$$

Filter gain calculation:

$$K(t_i) = P(t_i^-)C^T(t_i)\left[O(t_i)\right]^{-1} \quad (24)$$

with: $$O(t_i) = C(t_i)P(t_i^-)C^T(t_i) + R_d(t_i) \quad (25)$$

State estimate update:

$$\hat{x}(t_i^+) = \hat{x}(t_i^-) + K(t_i)\left[z(t_i) - C(t_i)\hat{x}(t_i^-)\right] \quad (26)$$

Error covariance update:

$$P(t_i^+) = \left[I - K(t_i)C(t_i)\right]P(t_i^-) \quad (27)$$

Steady State Solution. If the system and measurement dynamics are linear, constant coefficient equations and the noise is stationary (Q,R,S not functions of time), the filtering process will reach a steady state where the value of the error covariance, P, is constant. For these conditions the Riccati equation (equation 21) becomes an algebraic relationship:

$$\dot{P} = FP + PF^T + LQL^T - KRK^T = 0 \quad (28)$$

In this case, the rate at which uncertainty increases is just balanced by the new information available. The error covariance extrapolation and update and filter gain equations are no longer required. The positive semidefinite solution of the algebraic Riccati equation is used as the error covariance and to calculate the constant filter gain.

The discrete steady state solution error covariance extrapolations (equation 23) and filter gain (equation 24) are calculated from the following two equations:

$$\begin{aligned} K(\theta) &= \left[A(\theta)P(\theta)C^T(\theta) + S_d(\theta)\right] \\ &\quad \left[C(\theta)P(\theta)C^T(\theta) + R_d(\theta)\right]^{-1} \end{aligned} \quad (29)$$

$$\begin{aligned} P(\theta) &= A(\theta)P(\theta)A^T(\theta) + Q_d(\theta) - \\ &\quad \left[A(\theta)P(\theta)C^T(\theta) + S_d(\theta)\right] \\ &\quad \left[C(\theta)P(\theta)C^T(\theta) + R_d(\theta)\right]^{-1} \\ &\quad \left[A(\theta)P(\theta)C^T(\theta) + S_d(\theta)\right] \end{aligned} \quad (30)$$

Nonlinear Stochastic Prediction. If we want to explicitly consider system dynamics for nonlinear stochastic predictor models there are two options: a nonlinear state space model or a simulation model. For probabilistic models the nonlinear state space model is defined as:

$$x(t_{i+1}|\theta) = f(t, x(t_i), u(t_i), w(t_i), \theta) \quad (31)$$

$$y(t_i|\theta) = h(t, x(t_i), u(t_i), \nu(t_i), \theta) \quad (32)$$

A simulation model, not to be confused with a simulation as a system description, disregards the process noise and simulates $\hat{y}(t|\theta)$ by simulating a noise free model using actual inputs and $w(t_i) = \nu(t_i) = 0$.

Ito Stochastic Prediction. As reasonable as the linear stochastic model seemed, it is not completely suitable. Although other models may be derived from these Langevin type equations, the Markovian description is typically lost. With this loss, complete knowledge of the probability density functions is required to determine system properties. This information is usually not available.

In the development of the model, $w(t)$ has been considered as the derivative of a process with independent, stationary increments. Actually the term $w(\cdot,\cdot)$ is the hypothetical derivative of Brownian motion (or the Wiener process). A hypothetical derivative is used because the correct solution could not be properly developed with ordinary Riemann integrals.

Linear stochastic differential equations can be properly developed through the use of Wiener stochastic integrals [29]. Therefore the properly defined linear stochastic differential equation is:

$$dx(t) = F(t)x(t)dt + B(t)u(t)dt + G(t)d\beta(t) \quad (33)$$

where $\beta(\cdot,\cdot)$ is of diffusion strength $Q(t)$ for all t of interest given by $E\{d\beta(t)d\beta^T(t)\} = Q(t)dt$.

The solution to this stochastic differential equation is the stochastic process $x(\cdot,\cdot)$ given by:

$$\begin{aligned} x(t) &= \Phi(t,t_0)x(t_0) + \int_{t_0}^{t} \Phi(t,\tau)B(\tau)u(\tau)d\tau \\ &\quad + \int_{t_0}^{t} \Phi(t,\tau)G(\tau)d\beta(\tau) \end{aligned} \quad (34)$$

with $\Phi(t,t_0)$ the state transition matrix associated with F.

In general, characterization of this process requires the joint probability density (or distribution if the density cannot be assumed to exist) of $x(t_1), x(t_2), \ldots, x(t_N)$ for any number N of time cuts in the interval of interest by repeated application of Bayes rule. If $x(\cdot,\cdot)$ is a Markov process, however, specification of the transition probability densities completely specifies the joint densities and the transition probabilities can be propagated via the forward Kolmogorov equation.

Linear models. If the system model is (Markov) linear, solution to the forward Kolmogorov equation yields the familiar form of the state and covariance update:

$$\dot{\hat{m}}_x(t) = F(t)\hat{m}_x(t) \quad (35)$$

$$\dot{P}_x(t) = F(t)P_x(t) + P_x(t)F^T(t) + G(t)Q(t)G^T(t)$$

(Note: In the development of error criterion, etc. derivatives must be computed using the Ito differential rule.)

Nonlinear models. If we are willing to neglect the second partial derivatives with respect to x, we can use the extended Kalman filter [32]. Consider the general nonlinear model:

$$\dot{x}(t) = f\left[\hat{x}(t), u(t), t\right] + L(t)w(t) \tag{36}$$

with $x(t_0)$ modeled as a Gaussian random vector with mean x_0 and covariance P_0 and a measurement model of:

$$z(t_i) = h\left[\hat{x}(t_i), t_i\right] + \nu(t_i) \tag{37}$$

The extended Kalman filter for this model is the following set of equations:

State estimate extrapolation by integrating from t_i to t_{i+1}:

$$\dot{\hat{x}}(t/t_i) = f\left[\hat{x}(t/t_i), u(t), t\right] \tag{38}$$

Error covariance extrapolation by integrating from t_i to t_{i+1}:

$$\begin{aligned} \dot{P}(t/t_i) &= F\left[(t; \hat{x}(t/t_i)\right] P(t/t_i) + \\ & \quad P(t/t_i)F^T\left[(t; \hat{x}(t/t_i)\right] + G(t)Q(t)G^T(t) \end{aligned} \tag{39}$$

Filter gain calculation with $\hat{x}(t_i^+) = \hat{x}(t_i|t_{i-1})$:

$$\begin{aligned} K(t_i) &= P(t_i^-)H^T\left[t_i; \hat{x}(t_i^-)\right] \\ & \quad \left\{H\left[t_i; \hat{x}(t_i^-)\right] P(t_i^-)H^T\left[t_i; \hat{x}(t_i^-)\right] + \right. \\ & \quad \left. R(t_i)\right\}^{-1} \end{aligned} \tag{40}$$

State estimate update:

$$\hat{x}(t_i^+) = \hat{x}(t_i^-) + K(t_i)\left[z(t_i) - h\left[\hat{x}(t_i^-), t_i\right]\right] \tag{41}$$

Error covariance update:

$$\begin{aligned} P(t_i^+) &= \left\{I - K(t_i)H\left[t_i; \hat{x}(t_i^-)\right]\right\} P(t_i^-) \\ & \quad \left\{I - K(t_i)H\left[t_i; \hat{x}(t_i^-)\right]\right\}^T \\ & \quad + K(t_i)R(t_i)K^T(t_i) \end{aligned} \tag{42}$$

where the following definitions apply:

$$F\left[(t; \hat{x}(t/t_i)\right] = \left.\frac{\partial\left[x(t), u(t), t\right]}{\partial x}\right]_{x=\hat{x}(t/t_i)} \tag{43}$$

and

$$H\left[t_i; \hat{x}(t_i^-)\right] = \left.\frac{\partial h\left[x, t_i\right]}{\partial x}\right]_{x=\hat{x}(t_i^-)} \tag{44}$$

In the general case, the nonlinear problem is not solvable. There are a number of other approximations that exploit a Taylor series representation of the dynamics and measurement to estimate conditional moments. One of the more computationally reasonable is the modified Gaussian second order filter (Refer to [29]).

7.4 Summary of the Selection of the Model Set

Selection of the metamodel set is clearly defined by the system description, system class and a metamodel structure. Data for all of these selections come directly from the problem definition step.

Realization of the metamodel comes from the parameterization of the selected metamodel set. While the system description and system class constrain available representations (structures), a number of options are still available to the analyst. To help in this process we separated the decision into predictor and probabilistic models and within each of these general cases we presented a number of potential representations. In each case (predictor and probabilistic) the complexity of the model increases from linear time-invariant to general nonlinear.

The preferred solution is the simplest model structure that provides required performance. A decision not addressed in this paper is the order of the metamodel (This issue is discussed in [24]). If the selected structure (and order) does not provide the desired performance there are two options. First, the order of the model can be increased; secondly, a different (presumably more complex) metamodel structure can be selected. This selection must be driven by the nature of the performance shortfall and the degree of mismatch between the simulation model and the metamodel.

8 Selection of the Identification Methodology

We have selected a model set that we will use for the identification. We now discuss techniques for generating the estimate.

Parameter identification methods are used when the candidate model is to be defined by a set of parameters. Parameter estimation algorithms mentioned in the literature include least squares, sequential weighted least squares, recursive generalized least squares, instrumental variables, recursive instrumental variables, the bootstrap method, sequential correlation and recursive maximum likelihood estimation, etc. A partial list of algorithms included 32 different methods. Again, to structure the decision process, we classify these methods by two elements: the form of the identifier and the criterion of fit.

8.1 Form of the Identifier

The form of the identifier defines the "experimental setup" or the manner in which the estimates are generated and compared. The criterion of fit estab-

lishes both the cost function and the method of its minimization.

8.1.1 Equation Error Method

For the equation error method, Figure 3, we use the system equations as given. Assume first that we have the following general description defined by a parameter vector θ and that we know the form of the vector functions f and h:

$$\dot{x}(t) = f(t, x(t), u(t), w(t); \theta) \tag{45}$$

$$y(t) = h(t, x(t), u(t), \nu(t); \theta) \tag{46}$$

Now we assume that we can measure the controls, u_a, the states, x_a, and the state derivatives $\dot{x}_a$. With all of this information we can determine the error between the model and the actual data:

$$\varepsilon(t, \theta) = \dot{x}_a - f(x_a, u_a; \theta) \tag{47}$$

The vector $\varepsilon(t, \theta)$ is the equation errors. From these equation errors, $\varepsilon(t, \theta)$, we can form some nonnegative function such as $J(\theta) = \int_0^T \varepsilon^T(t, \theta)\varepsilon(t, \theta)dt$ and search over θ to find the minimum.

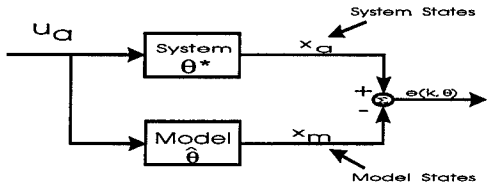


Figure 3: Equation Error Method

8.1.2 Output Error Method

The equation error method required measurement of all of the elements of the system. Often this is not possible. The output error method is based on an output error criterion and avoids this requirement. Figure 4 depicts the experimental setup for the output error method. As you see, there is no attempt to measure the state of the plant. Instead the estimated parameter, θ, is used in the model with the input u_a to generate an estimate of the output y_m. Again we can form some nonnegative function of the difference between y_m and y_a.

8.1.3 Prediction Error Method

The prediction error method is the third approach to developing an error function by which a parameter search can be structured (Figure 5). Instead of comparing states or outputs the estimated parameter, θ, is used in the model with the input u_a and the output y_a to generate an estimate of the output y_m. Given a description:

$$y(t) = G(q)u(t) + H(q)e(t) \tag{48}$$

and having observed the output y and the input u, the prediction errors can be computed as:

$$e(t) = H^{-1}(q)\left[(y(t) - G(q)u(t)\right] \tag{49}$$

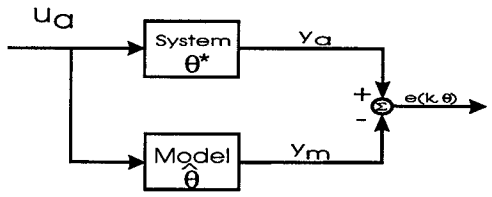


Figure 4: Output Error Method

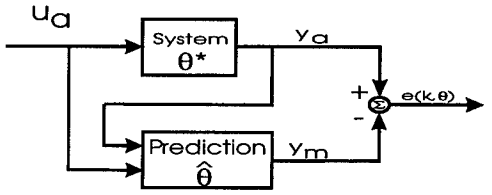


Figure 5: Prediction Error Method

8.2 Criterion of Fit

So far, we have defined the metamodel set that will be parameterized to generate the metamodel, the form of the identifier that describes how the data is generated and compared. Now we discuss the criterion of fit.

By criterion of fit we mean the function or functional that is optimized to determine the parameter estimates.[8].

Our framework is based on the assumption that the data will contain both measurement errors and system disturbances not accounted for by the model. Consequently, measurements are realizations of a stochastic system and are represented by functions of random variables that have some probability density function.

The probability that a particular random variable is in the range $a \leq x_i \leq b$ is given by: $Pr\{a \leq x_i \leq$

[8] We do not know the actual parameter vector θ_* and cannot define an error between θ_* and $\hat{\theta}$. The error must be computed from $\{z(t_i)\}, \{u(t_i)\}$, and $\{y(t_i)\}$

$b\} = \int_a^b f_{x_i}(\xi)d\xi$ where $f_{x_i}(\xi)$ is the probability density function (PDF) of the set of $\{x_i\}$. Therefore, the PDF is a measure of the "likelihood" of a particular value.

Assume that the PDF of the measurements Z^N is $f(\theta; z_1, z_2, \ldots, z_N) = f_z(\theta; Z^N)$ where θ is a d-dimensional parameter vector determined by the parameter estimator. This PDF is a joint PDF (JPDF) that considers the joint (combined) probability of both θ and z_N occurring. However, because we have a function of a random variable and measurements that are available in a specific sequence, we can also consider the conditional probability distributions (CPDF). That is, the probability of an event conditioned on the fact that another event has occurred such as $P(z|\hat{\theta})$ which is the probability of a particular z_N conditioned on the fact that $\theta = \hat{\theta}$.

It is entirely possible that an identification method, given a model and a particular set of data, has multiple characteristics. For example, least squares is a specific case of the prediction error method that minimizes a norm of the prediction error. Yet, if the data meets the assumptions of the method, least squares is also a maximum likelihood estimator since it also maximizes the likelihood of the parameter vector given the observations $f_z(\theta; Z^N)$.

We consider three criterion: minimum mean square, maximum a posteriori (maximize the CPDF) and maximum likelihood (maximize the JPDF).

8.2.1 Minimum Mean Square Error

Minimum mean square estimators minimize a cost function that is a function of the (possibly weighted) output error only - $J(\hat{\theta}) = \varepsilon^T W \varepsilon$. The mean square error matrix MSE for an estimate of $\hat{\theta}$ of θ (with b equal to the bias) is:

$$MSE = E\left\{(\hat{\theta} - \theta)(\hat{\theta} - \theta)^T\right\} = cov\hat{\theta} + bb^T \qquad (50)$$

Both bias and covariance must both be minimized to attain the minimum mean square estimate; and in general, the minimum m.s.e. will be biased. The minimum m.s.e. estimator will, however, result in output errors (residuals) that are orthogonal to the estimate.

8.2.2 Maximum A-Posteriori

The Bayesian approach to parameter estimation assumes a parameter vector with *a priori* (before the measurement) probability densities $P(\theta)$. The observations Z^N are therefore correlated with θ. Measurements are used to determine the most likely value after the measurement, the Maximum A-Posteriori (MAP) estimate $\hat{\theta}_{MAP}$ via the application of Bayes rule:

$$P(\hat{\theta}|z) = \frac{P(z|\hat{\theta}) \times P(\hat{\theta})}{P(z)} \qquad (51)$$

Here $P(z|\hat{\theta})$ is the conditional probability; i.e., the total probability of the measurement conditioned on the current estimate of θ.

We can rewrite the maximization to be the minimization of the negative logarithm of $P(z|\hat{\theta})$:

$$\hat{\theta}_{MAP} = arg(\hat{\theta}) \min_{\theta} \left[-\log P(\hat{\theta}|z)\right] \qquad (52)$$

where $\log P(\hat{\theta}|z) = \log P(z|\hat{\theta}) + \log P(\hat{\theta}) - \log P(z)$. Since $P(z)$ is unaffected by $\hat{\theta}$, it can be ignored in the minimization.

Assuming the correct *a priori* probability, the MAP estimate minimizes the $E\left\{(\hat{\theta} - \theta)(\hat{\theta} - \theta)^T\right\}$ and, therefore, is the minimum-quadratic-cost estimate. The MAP estimate also minimizes the expected absolute error $E\left\{(\hat{\theta} - \theta)|Z^N\right\}$.

8.2.3 Maximum Likelihood

Given that the joint probability of the random vector to be observed is $f_z(\theta; Z^N)$, the probability that the random variable will produce the realization Z^N is proportional to $f_z(\theta; Z^N)$. Once a particular realization Z_*^N is inserted into the joint PDF, this becomes deterministic and is called the likelihood function. A maximum likelihood estimator maximizes this function:

$$\hat{\theta}_{ML} = arg(\theta) \max_{\theta} f_z(\theta; Z^N) \qquad (53)$$

so that the observed event becomes as likely as possible.

Beginning with the MAP estimate and ignoring the prior information, we have for the ML estimate $\hat{\theta}_{ML} = arg(\hat{\theta}) \min_{\hat{\theta}} \left[-\log P(z|\hat{\theta})\right]$.

Comparing the ML and MAP log likelihood functions (LLFs), we see that $LLF_{MAP} = LLF_{ML} + \log P(\hat{\theta})$.

While the MLE has been criticized for poor small sample properties, the statistical properties of maximum likelihood estimators for a "sufficiently long" data sample are [16]:

1. Parameter errors have an unbiased Gaussian distribution.
2. Estimates are consistent - unbiased as the data length increases.
3. Efficient estimates - no unbiased estimator has lower error variance.

8.3 Summary of the Identification Methodology

Referring back to Figure 1, we have defined the problem in terms of the use of the metamodel and the simulation that will be modeled. We used that information along with the data that came from the simulation to select the metamodel set that will be parameterized to realize the metamodel. We then presented three methods and three criterion that can be used to generate the data and provide the "cost" function for optimization.

Categorizing the identification method by the form and the criterion reduces the myriad of identification methods to only four approaches: Prediction Error and Correlation, Maximum Likelihood, Optimization and Approximation Techniques.

9 Generate the Metamodel

There are many taxonomies used in the literature to categorize identification methods. Methods can be referred to as off-line or on-line. Also, they can be classified as either open-loop or closed-loop methods. Further classification can be made as nonparametric, frequency domain and as parameter identification methods. As stated, we have reduced the parameters identification methods to four approaches. We will now present some of the techniques that result. A summary discussion of these elements is included in [34]. Additional details are found in [24].

Assume that a model structure (set of candidate models) has been selected and parameterized using some parameter vector θ. We have defined the model set $M(\theta)$. The next step is to search for the best model within the set (determine the parameter vector θ). The objective is to determine the most powerful unfalsified model (MPUM) where a model is the MPUM based on the data D if: (1) $M \in M(\theta)$; (2) $M(\theta)$ is unfalsified by D; and (3) $M(\theta)$ is more powerful than any other model satisfying (1) and (2). We must determine the mapping from the data set D to $M(\theta)$.

9.1 Prediction Error and Correlation Approaches

Let the prediction error be given by $\varepsilon(t,\theta) = y(t) - \hat{y}(t|\theta)$ with $y(t)$ the output of the simulation and $\hat{y}(t|\theta)$ the output of the metamodel (θ is the parameter vector). A "good" model will have small prediction errors. There are two general approaches to define a measure of ε. The first is to define a norm that measures the size of ε and minimize that norm. This leads to the prediction error method (PEM). Another measure of ε is to require that ε be uncorrelated with past data. This is the correlation approach which contains the instrumental-variable (IV) method which we discuss in Section 9.1.3.

In addition to least squares (LS), subsets of the prediction error method also include the maximum likelihood approaches (ML and MAP). Our discussion separates out the maximum likelihood approaches from PEM. We do so because when we consider probabilistic models (where ML and MAP estimators apply) the prediction equations for explicitly using the PEM algorithm are limited to the directly parameterized form. There are a number of other probabilistic model structures where the PEM algorithm cannot be used.

We do include, however, the Eigenstructure Realization Algorithm (ERA) under PEM. We do so because this algorithm uses the least squares approach to directly identify the Markov parameters of a steady state Kalman filter.

9.1.1 General Description of The Prediction Error Method

Filter the prediction sequence $\varepsilon(t,\theta)$ using a stable linear filter $L(q)$:

$$\varepsilon_F(t,\theta) = L(q)\varepsilon(t,\theta) \tag{54}$$

where q is the forward shift operator defined as $qu(t) = u(t+1)$.

This filtering acts like frequency weighting and can remove or enhance selected properties of the model. Using either a fixed or weighted (possibly time varying) norm $l(\cdot)$:

$$V_N(\theta, D) = \frac{1}{N}\sum_{t=1}^{N} l\left(\varepsilon_F(t,\theta),\theta,t\right) \tag{55}$$

define the estimate $\hat{\theta}_N$ by the minimization:

$$\hat{\theta}_N = \hat{\theta}_N(D) = arg\min_{\theta \in D}\{V(\theta, D)\} \tag{56}$$

where D is the set allowed by the model.

In general PEM is a technique that approximates (smoothes) the empirical transfer function estimate to the model transfer function with a weighted norm corresponding to the model signal-to-noise at the frequency in question.

9.1.2 Specific PEM Methods

While "equation 56" can be solved numerically in the general case, specific methods are obtained as special

cases with special selections of the filter $L(q)$ and the scalar valued norm function $l(\cdot)$.

Least Squares. If the predictor is linear, the prediction error becomes $\varepsilon(t,\theta) = y(t) - \phi^T(t)\theta$ where $\phi^T(t)$ is the vector of regressors that depends on the selected model structure. Also if $L(q) = 1$ and $l(\varepsilon) = \frac{1}{2}\varepsilon$, then the norm becomes:

$$V_N(\theta, D) = \frac{1}{N}\sum_{t=1}^{N}\frac{1}{2}\left[y(t) - \phi^T(t)\theta\right]^2 \qquad (57)$$

This is the least squares criterion for linear regression. The performance measure $J = \varepsilon^T\varepsilon$ was based on the view that all errors are equally important [19]. Weighted least squares weights the errors and is based on the criterion $J = \varepsilon^T W\varepsilon$. Other versions of the least squares criterion are the Best Linear Unbiased Estimator where the weight is equal to the inverse of the measurement noise [35].

If the variance of the parameters is known (or assumed), we can further improve on the Best Linear Unbiased (Gauss-Markov) Estimator. This improvement is called the minimum variance estimator and includes the variance of the parameters in the normal equations [20].

Ridge Regression. The aim of another modification of ordinary least squares - ridge regression - is the reduction of the mean square error [17]. This is accomplished by the addition of a symmetric matrix K to the regressor to improve the numerical conditioning of the estimator.

Chi-Square. In Chi-Square identification we assume that each data point y_i has a measurement error that is independently random and distributed as a normal distribution around the true model. Suppose the standard deviation is the same for all points; it follows that the probability of the data set is the product of probabilities of each point:

$$P = \Pi\{\exp\left[-\frac{1}{2}\left(\frac{y_i - y(x_i)}{\sigma}\right)^2\right]\Delta y\} \qquad (58)$$

Maximizing this is equivalent to maximizing its logarithm, or minimizing the negative of its logarithm:

$$\left[\sum\frac{[y_i - y(x_i)]^2}{2\sigma^2}\right] - N\log\Delta y \qquad (59)$$

Since N,σ, and Δy are all constants, minimizing this equation is equivalent to minimizing:

$$\sum_{i=1}^{N}[y_i - y(x_i;\theta_1\ldots\theta_M)]^2 \qquad (60)$$

If each data point has its own standard deviation the probability of the data set is modified by considering σ_i in place of σ (Refer to [17] for details).

Eigenstructure Realization Algorithm. The Eigenstructure Realization Algorithm (ERA) is included under the PEM methods because this algorithm uses the least squares approach to directly identify the Markov parameters of a steady state Kalman filter.

Consider a discrete, time-invariant multivariable linear system:

$$\begin{aligned} x_{t_{i+1}} &= A(\theta)x(t_i) + B(\theta)u(t_i) + M(\theta)w_d(t_i) \\ y(t_i) &= C(\theta)x(t_i) + D(\theta)u(t_i) + \nu(t_i) \end{aligned} \qquad (61)$$

An observer for the above system can be developed that will be as stable as desired and the resulting Markov parameters will be the Markov parameters of the observer. The system Markov parameters can be extracted from the observer parameters. The major assumption is that of ergoticity.

Choose p such that $mp > n$ (where n is the number of states and m is the number of outputs) and, beginning at the p+1 measurement, let:

$$y = [y(p+1)\; y(p+2)\; y(p+3)\ldots y(k-1)] \qquad (62)$$

From the definition of the Kalman Filter we have:

$$\overline{Y} = \left[\, D\; C\overline{B}\; C\overline{AB}\ldots C\overline{A}^{k-1}\overline{B}\,\right] \qquad (63)$$

with

$$\begin{aligned} \overline{A} &= A + MC \\ \overline{B} &= [B + MD, -M] \end{aligned}$$

and if

$$U = \begin{vmatrix} u(p+1) & u(p+2) & \cdots & u(k-1) \\ \begin{bmatrix} u(p) \\ y(p) \end{bmatrix} & \vdots & \cdots & \begin{bmatrix} u(k-2) \\ y(k-2) \end{bmatrix} \\ \vdots & \begin{bmatrix} u(p) \\ y(p) \end{bmatrix} & \cdots & \begin{bmatrix} u(k-3) \\ y(k-3) \end{bmatrix} \\ \vdots & \ddots & \vdots & \vdots \\ \begin{bmatrix} u(0) \\ y(0) \end{bmatrix} & \begin{bmatrix} u(1) \\ y(1) \end{bmatrix} & \cdots & \begin{bmatrix} u(k-p-2) \\ y(k-p-2) \end{bmatrix} \end{vmatrix} \qquad (64)$$

When $C\overline{A}^k\overline{B} \approx 0$ for $k > p$, the system $y = \overline{Y}U$ can be solved for $\overline{Y}$ using a weighted least squares. Once the observer Markov parameters are determined the system parameters must be extracted. After extracting the system Markov parameters from the observer, we can recover the state space model by the ERA. Define the following $r_1 \times s$ block data matrix:

$$H(\tau) = \begin{bmatrix} Y_\tau & Y_{\tau+1} & \cdots & Y_{\tau+s-1} \\ Y_{\tau+1} & Y_{\tau+2} & \cdots & Y_{\tau+s} \\ Y_{\tau+2} & Y_{\tau+3} & \cdots & Y_{\tau+s+1} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{\tau+r_1-1} & Y_{\tau+r_1} & \cdots & Y_{\tau+r_1+s-2} \end{bmatrix} \quad (65)$$

The order of the system is determined by the singular value decomposition of $H(0)$:

$$H(0) = U\Sigma V^T = U_1 S_1 V_1^T \quad (66)$$

where Σ are all of the singular values. S_1 is an $n \times n$ diagonal matrix of positive singular values that are retained and n will become the order of the system:

$$\begin{aligned} A &= S_1^{-1/2} U_1^T H(1) V_1 S_1^{-1/2} \quad (67) \\ B &= S_1^{-1/2} V_1 E_m \\ C &= E_1^T U_1 S_1^{-1/2} \end{aligned}$$

where $E_r^T = \left[I_{r\times r}\ 0_{r\times(r_1-m)m}\right]$ and $E_m^T = \left[I_{m\times m}\ 0_{m\times(r_1-m)m}\right]$. The observer gain can be extracted in a similar fashion. See [36] for the details of this method.

9.1.3 Correlation Approaches

Ideally the prediction error $\varepsilon(N,\hat{\theta})$ for a "good" model should be independent of past data Z^{N-1}. If $\varepsilon(N,\hat{\theta})$ is correlated with past data there is more information available in the data. A true test of the correlation of $\varepsilon(N,\hat{\theta})$ and Z^{N-1} requires testing every nonlinear transformation of $\varepsilon(N,\hat{\theta})$ with all possible functions of Z^{N-1}. This is not feasible.

We can, however, select a finite dimensional vector sequence $\{\zeta(t)\}$ derived from Z^{N-1} and force a certain transformation of $\varepsilon(N,\hat{\theta})$ to be uncorrelated with this sequence. In general, we can accomplish this by filtering the prediction errors:

$$\varepsilon_F(N,\hat{\theta}) = L(q)\varepsilon(N,\hat{\theta}) \quad (68)$$

choosing a sequence of correlation vectors:

$$\{\zeta(t,\hat{\theta})\} = \{\zeta(t)(t, Z^{N-1}, \hat{\theta})\} \quad (69)$$

and a function: $\alpha\left(\varepsilon_F(N,\hat{\theta})\right)$ for computing:

$$f_N(\hat{\theta}, Z^{N-1}) = \frac{1}{N}\sum_{t=1}^{N} \zeta(t,\hat{\theta})\alpha\left(\varepsilon_F(N,\hat{\theta})\right) \quad (70)$$

and then finding $\hat{\theta}_N$ such that $f_N(\hat{\theta}, Z^{N-1}) = 0$.

If we define $\varepsilon(N,\hat{\theta})$ above to be $\varepsilon(N,\hat{\theta}) = \left[y(t) - \phi^T(t)\hat{\theta}\right]$, we can expand the sequence of the correlation vectors to include model dependent parameters by:

$$\{\zeta(t,\hat{\theta})\} = K_u(q,\hat{\theta})u(t) \quad (71)$$

where $K_u(q,\hat{\theta})$ is a $d \times m$ matrix filter and $L(q)$ is of dimension $p \times p$. With $dim\zeta(t) = dim\hat{\theta} = d \times p$, we have the instrumental-variable (IV) method:

$$\hat{\theta}_{IV} = \left[\zeta(t,\hat{\theta})^T X\right]^{-1} \zeta(t,\hat{\theta})^T y \quad (72)$$

If we allow $dim\zeta(t) > d$ and a minimum norm solution for $f_N(\hat{\theta}, Z^N)$, we have the extended IV method. (Reference [16] discusses this method in detail.)

9.2 Maximum Likelihood Approaches

If we consider independent, identically distributed measurements and if an efficient estimate (unbiased estimate with finite covariance such that no other unbiased estimate has a lower covariance) exists, it can always be found through maximum likelihood approaches. Again if an efficient estimate exists, the likelihood equation will have a unique solution that equals the efficient estimate. If any single sufficient statistic exists, the maximum likelihood estimate will be sufficient. Although the maximum likelihood estimate will be biased for small samples it will provide the unique minimum variance estimate attaining the Cramér-Rao lower bound if this is possible [29].

The objective is to provide a parameter estimator that does not require complete *a priori* parameter statistics yet still allows the inclusion of *a priori* knowledge. Unlike the best linear unbiased estimate provided by appropriately weighted least squares, this method propagates the probabilistic information in time and directly allows the inclusion of known statistical information.

The key to the identification algorithm will be the residuals of the state estimator and the most significant drawback of the maximum likelihood approaches is the lack of theoretical knowledge on the behavior of the estimates for small sample sizes.

The following discussions are limited to linear-time invariant (discrete time) systems. Nonlinear effects can be included by appropriately modifying the prediction equations in either of two ways. First, nonlinear system effects can be directly included in the propagation of the state. Second, nonlinear measurements (with linear propagation) can be handled with an extended Kalman filter model.

Beginning with a linear time-invariant discrete state space model (equation 61) there are a number of conditional probability destiny functions that could be used for the likelihood function. Variations include

fixed length versus growing length functions, specification of *a priori* statistics, use of the initial conditions and the sensitivity of the estimate on the identified parameters. The most appropriate density function is:

$$\begin{aligned} f_{x(t_i),Z(t_i)|\theta} &= f_{x(t_i)|Z(t_i),\theta} f_{Z(t_i)|\theta} \\ &= f_{x(t_i)|Z(t_i),\theta} \Pi_{j=1}^{i} f_{x(t_i)|Z(t_i),\theta} \end{aligned} \quad (73)$$

Minimization of the likelihood function with this density results in the state predicted by the Kalman-Bucy filter, but there is no closed form solution to compute the partial derivatives.

9.2.1 Full Scale Estimator

A full scale estimator can be derived that minimizes the likelihood function in an iterative process. This estimator uses the last N observations to identify v uncertain parameters in the system and input matrices A and B. (Note: Uncertainty in these parameters could not be separated from uncertainties in C and D. Consequently, the assumption is that C and D are known and the uncertainty is A and B.)

The iterative estimator for minimization of the likelihood equation:

$$\left. \frac{\partial L\left[\hat{\theta}, Z^N\right]}{\partial \hat{\theta}} \right|_{\hat{\theta}(t_i)=\theta_*(t_i)} \quad (74)$$

using the method of "steepest descent" is:

$$\hat{\theta}(t_i) = \hat{\theta}(t_i) - \left[\frac{\partial^2 L\left[\hat{\theta}, Z^N\right]}{\partial \hat{\theta}^2}\right]^{-1} \left[\frac{\partial L\left[\hat{\theta}, Z^N\right]}{\partial \hat{\theta}}\right] \quad (75)$$

To use this algorithm, the Hessian (second derivative matrix) must be of full rank. Using a technique called "scoring," we can approximate the Hessian with the conditional information matrix. However, considering the propagation of the values in time, incorporation of measurements and the summation over the last N residuals, the implementation of the above equations is quite complex. Even with the approximations the full scale estimator requires a large number of calculations (Refer to [29]).

9.2.2 Modified Maximum Likelihood (MMLE)

The modified maximum likelihood formulation uses a discrete state variable representation (equation 61) where A, B, C, D, and M are estimated and used with the error covariance, P, to determine the Kalman gain, K, from an approximation based on the Ricatti equation [37]. To provide a parameter estimator we consider the measurement equation. Since we have assumed a Gaussian error model, the Conditional Probability Density Function (CPDF) for the measurement becomes:

$$P(z_i|z_{i-1},\theta) = \frac{1}{[(2\pi)^m det P]^{1/2}} \exp\left\{-\frac{1}{2}\tilde{z}_i^T (P)^{-1} \tilde{z}_i\right\} \quad (76)$$

where $P = E\left\{\tilde{z}\tilde{z}^T\right\}$ with dimension $m \times m$ and $\tilde{z} = z_i - \hat{z}$ is the innovations process (residuals) computed by the Kalman filter (where all of the matrices could be functions of θ).

Assuming a constant innovations covariance, use of a steady state filter results in a constant filter gain. This allows the CPDF to be written as:

$$P(z|\theta) = \Pi_{i=1}^{N} \frac{1}{[(2\pi)^m det P]^{1/2}} \exp\left\{-\frac{1}{2}\tilde{z}_i^T (P)^{-1} \tilde{z}_i\right\} \quad (77)$$

There are two approaches to the solution depending on whether *a priori* information is used.

Maximum Likelihood (ML) Estimation. Given the above CPDF, the ML LLF becomes:

$$\begin{aligned} LLF(\hat{\theta}) &= \frac{1}{2}\sum_{i=1}^{N}\left\{\tilde{z}_i^T (P)^{-1} \tilde{z}_i\right\} \\ &\quad + \frac{N}{2}\log\det(P) + \frac{Nm}{2}\log 2\pi \end{aligned} \quad (78)$$

A necessary condition at the minimum is that $P = E\left\{\tilde{z}\tilde{z}^T\right\}$ must equal the sample innovations covariance [38]. Therefore since P has dimension $m \times m$, the first term in the LLF becomes $Nm/2$, and the minimization is reduced to a minimization of the determinant of the sample innovations covariance matrix.

When P is known the LLF can be minimized by minimizing the following cost function:

$$J(\hat{\theta}) = \frac{1}{2}\sum_{i=1}^{N}\left\{\tilde{z}_i^T (P)^{-1} \tilde{z}_i\right\} \quad (79)$$

This minimization is usually carried out by the Gauss-Newton method using the first and second gradients of the cost function.

Maximum A Posteriori (MAP) Estimation. In the MAP estimator, we continue to require that $\hat{P} = \frac{1}{N}\sum_{i=1}^{N} \tilde{z}\tilde{z}^T$ but add the term $-\log P(\hat{\theta})$.

Assuming that θ is normally distributed with a covariance Σ:

$$\begin{aligned} -\log P(\hat{\theta}) &= \frac{1}{2}\left(\hat{\theta}-\theta\right)^T \Sigma^{-1}\left(\hat{\theta}-\theta\right) + \\ &\quad \frac{1}{2}\log\left((2\pi)^m \det\Sigma\right) \end{aligned} \quad (80)$$

the LLF_{MAP} becomes:

$$LLF_{MAP}(\hat{\theta}) = \frac{1}{2}\sum_{i=1}^{N}\left\{\tilde{z}_i^T(\hat{P})^{-1}\tilde{z}_i\right\} + \frac{1}{2}(\hat{\theta}-\theta)\Sigma^{-1}(\hat{\theta}-\theta) \quad (81)$$

which adds a quadratic term that biases the estimates toward *a priori* values.

9.3 Optimization

Often we are unable to formulate the problem to achieve a suitable prediction equation. Therefore we must resort to either a "nonlinear state space model" or a "simulation model." In these situations, where we are unable or unwilling to consider a linearized or perturbation approach, the best we can do is take the output of the model, incorporate it into a "cost" function, and adjust the model parameters to optimize (minimize) that function.

There are several "standard" numerical procedures that are used to search for the minimum of a function. These are the iterative optimization methods: Successive approximation, Newton's method, or the Gauss-Newton algorithm to name a few.

In addition there are several programs that are specifically designed to perform parameter estimation.

pEst. A minimum mean square error parameter estimator, pEst is an interactive program for the parameter estimation of nonlinear dynamic systems [39]. This program solves a vector set of time-varying, finite-dimensional, ordinary differential equations that are separated into a continuous-time state equation and a discrete-time measurement equation:

$$\begin{aligned}\dot{x} &= f[x(t), u(t), \theta] \\ z(t_i) &= g[x(t_i), u(t_i), \theta]\end{aligned} \quad (82)$$

pEst uses three separate minimization algorithms (steepest descent, modified Newton-Raphson and Davidod-Fletcher-Power) to minimize the following weighted cost function:

$$J(\hat{\theta}) = \frac{1}{2n_N n_z}\sum_{i=1}^{n_t}[z(t_i)-\hat{z}(t_i)]^T W[z(t_i)-\hat{z}(t_i)] \quad (83)$$

where n_N equals the number of data points, and n_z is the number of response variables.

Simulated Annealing. Using statistical mechanical theories an optimization technique called "simulated annealing" provides a new option to directly process nonlinear, discontinuous, stochastic functions [40]. Given data and a cost function, it will globally optimize that function by emulating the physical annealing process to arrive at a global minimum. (Reference [4] and [41] provide a description on how to use Adaptive Simulated Annealing.)

9.4 Approximation Techniques for Identification

9.4.1 Stochastic Approximation

Stochastic approximation may be regarded as the application of gradient methods to stochastic problems. It is a scheme for successive approximation of a sought quantity when the observations involve random errors due to the stochastic nature of the problem. The main advantage is the simplicity of the implementation and the fact that prior knowledge of the noise statistics are not necessary.

Stochastic approximation can be applied to any problem which can be formulated as a regression in which repeated observations are made. This approach is an exact analog of the deterministic gradient procedure.

9.4.2 Spline Approximation

Polynomials are excellent approximating functions when a smooth function is to be approximated locally. Any such smooth piecewise polynomial function is called a spline and they are commonly used for fitting data.

The typical use for the spline approximation is to construct a piecewise polynomial to fit data. An exact fit involves interpolation; an approximate fit uses least squares (minimum mean square error) approximation. To explain the structure and advantages of the spline, consider a truncated Taylor series (expanded about x_0 where D^i is the i^{th} derivative):

$$\sum_{i=0}^{n}\frac{(x-x_0)^i}{i!}D^i f(x_0) \quad (84)$$

This polynomial should provide a satisfactory approximation for $f(x)$ if the function is sufficiently smooth and x is sufficiently close to x_0. If the function must be approximated over a larger interval, the degree of the polynomial may have to be unacceptably large.

The alternative to a higher order polynomial is to subdivide the interval into sufficiently small intervals in order that, on each interval, a polynomial with a relatively low degree can provide an adequate approximation.

The construction of a series of splines over an interval is a stable and straightforward mathematical procedure [42]. At the breakpoints, derivatives are continuous. At the end points two conditions are possible. In the "natural" cubic spline the second derivative is zero. In the "not-a-knot" end condition the jump in the third derivative is zero.

Once developed, the spline can be evaluated, integrated, differentiated, augmented or cut.

9.4.3 Canonical Variate Analysis

Another approximation technique is canonical variate analysis. The canonical variate method is a prediction error approximation technique that optimally predicts future responses based on a reduced order state space system [43].

In the statistical literature the canonical variate problem is one of maximizing the correlation between two sets of variables. Here we will use the technique to choose nonlinear combinations of past data to predict the future data by considering the fact that the conditional expectation is an optimal projection in Hilbert space. We optimally select k linear combinations of the past data for prediction of the future.

Observations coming from the behavior we desire to model are separated into the past p(t) of a vector process and the future f(t) of another vector process. They are assumed to be jointly stationary:

$$\begin{aligned} p^T &= (y^T(t), y^T(t-1), \ldots, u^T(t), \ldots)^T \\ f^T &= (y^T(t+1), y^T(t+2), \ldots, y^T(t+l))^T \end{aligned} \quad (85)$$

where the vector process $p(t)$ can include both inputs and outputs.

The optimal k^{th} order linear predictor $\hat{f}(t)$ of the past is measured by the prediction error:

$$E\left\{\| f - \hat{f} \|^2_{\Lambda^{-1}}\right\} \equiv \left\{(f-\hat{f})^T \Lambda^{-1} (f-\hat{f})\right\} \quad (86)$$

where Λ is arbitrary positive semidefinite, so that Λ^{-1} is a quadratic weighting matrix that is possibly singular. The CVA problem is to determine $c(t) = J_k p(t)$ and $d(t) = L_k f(t)$ such that the prediction error is minimized. Each of the terms $c(t)$ and $d(t)$ are combinations of k terms and are defined in a new basis. The algorithm uses the properties of stochastic independence to find a canonical form of $c(t)$ and $d(t)$. This in turn provides J_k and L_k which can be used to predict system performance based on the past data.

The connection between CVA and metamodeling is not direct and much of the literature is very confusing or misleading. First recall that the metamodel is a reduced order model that is the result of an optimal projection of the higher order model onto a subspace of reduced dimensions. It can be shown that projection operators on a Hilbert Space of nonlinear functions can be expressed as a conditional expectation [43]. It can also be shown that eigenvectors of this conditional expectation have a common eigenvalue which is equal to the squared maximal correlation. If a process has a rational power spectrum (i.e. it is a finite order Markov process) there are a finite number of nonzero canonical correlations between the past and future outputs [44].

The solution to the canonical variate problem is expressed by putting the covariance structure of the past and future data in a canonical form such that in this new basis the norm of the weighted prediction error is the sum of squares. This is equivalent to finding J and L such that:

$$\begin{aligned} J\Sigma_{pp}J^T &= I_m \\ L\Lambda L^T &= I_n \end{aligned} \quad (87)$$

$$J\Sigma_{pf}L^T = Diag\left\{\gamma_1 \geq \gamma_2 \geq, \ldots, \geq \gamma_q \geq 0, \ldots, 0\right\} \quad (88)$$

where $\Sigma_{pp}, \Sigma_{ff}, and \Sigma_{pf}$ are the covariance matrices of past, future and cross covariance of the past and future data defined by:

$$\Sigma = \begin{pmatrix} \Sigma_{pp}\Sigma_{pf} \\ \\ \Sigma_{fp}\Sigma_{ff} \end{pmatrix} \quad (89)$$

with $Diag\left\{\gamma_1 \geq \gamma_2 \geq, \ldots, \geq \gamma_q \geq 0, \ldots, 0\right\}$ a diagonal matrix with the singular values on the diagonal. Since the past and future basis in the new basis are orthonornal and uncorrelated the singular values are also the correlations between the canonical variates p and f.

In a linear system, independent variables are orthogonal. For nonlinear systems stochastic independence is required. The maximal correlation is defined by:

$$\rho(p,f) = \sup_{p,f} \rho\left(p(y), f(y)\right) = \sup_{p,f} E\left\{p(y), f(y)\right\} \quad (90)$$

with $\| p \| = 1$ and $\| f \| = 1$.

If $\rho(p,f) = 0$, then $p(y), f(y)$ are statistically independent. Therefore to find the optimal combination of past data to predict the future we want the maximal correlation.

Determining this structure requires multiple steps. First, given the past and future vectors, the mean is removed to meet the constraints of the alternating conditional expectation (ACE) algorithm that will be used to determine the maximum correlation between transformed input and output variables c and d [45]. Then a (Σ_{pp}, Λ) singular value decomposition of Σ_{pf} will determine a J and L such that after the transformations $c(t) = J_k p(t)$ and $d(t) = L_k f(t)$ and the covariances $\Sigma_{cc} = \Sigma_{dd} = I$.

10 Results and Conclusions

In this paper we presented a new approach where we did not try to determine the best polynomial fit to a set of input-output data, but concentrated on the identification of the underlying systems that defined the process.

By focusing on the system theoretic properties of the manifest behavior we generated the metamodel via solution of a general inverse problem that did not restrict the solution to an approximation of the input-output map. This approach expanded the available classes of metamodels by supporting the development of dynamical models that incorporate memory. This expansion allowed the generation of metamodels that included system dynamics so that metamodels can be developed where the past could influence the future.

In addition to a new approach to the definition of the problem we presented a new framework for the solution. The framework centered on the behavior of the system, the behavioral equations that specified the behavior and latent variables which may have been present from first principles.

A structured metamodeling method was presented that simplified the metamodeling process to two phases: problem definition and the metamodeling process. In the problem definition we began with an analysis of the metamodel requirements and the simulation under study. We then progress to the description of the system (not the model) so that we will be able to select a metamodel structure that matches both the requirements and simulation that we are going to metamodel.

The structured metamodeling method segmented the metamodeling process into a set of sequential decisions: Definition of the Problem; Selection of the Metamodel Set; Selection of the Identification Methodology; and Generation of the Metamodel. In each case we step through decisions that are based on existing information or follow from prior decisions. We have added the capability to explicitly model dynamical systems and defined the requirements to use these as metamodels.

This structured metamodeling method was supported by new taxonomies of metamodel structures (representations), identification methodologies and methods to generate the metamodel that allowed separation of the metamodeling process into a set of sequential decisions based on *a priori* information.

References

[1] M. A. Zeimer, et. al., "Metamodel Procedures for Air Engagement Simulation Models," *IRAE Technical Report*, Jan 1993.

[2] A.F. Sisti, "Large-Scale Battlefield Simulation Using a Multi-Level Model Integration Methodology," *RL-TR-92-69*, April 1992.

[3] D. Caughlin, "A Metamodeling Approach to Model Abstraction," *Proc. 1994 Fourth Annual IEEE Dual Use Technologies and Applications Conference*, May 1994.

[4] D. Caughlin, "An Evaluation of Simulated Annealing for Modeling Air Combat Simulations," Proc. 1994 IEEE Dual-Use Technologies and Application Conference, May 1994.

[5] J. C. Willems, "Paradigms and Puzzles in the Theory of Dynamical Systems," *IEEE Trans. on Automat. Contr.*, vol. 36, no. 3, pp. 259-294, March 1991.

[6] D. Caughlin, "Verification, Validation, and Accreditation (VV&A) of Models and Simulations Through Reduced Order Metamodels," *Proc. 1995 Winter Simulation Conference*, December 1995.

[7] V. Vemuri, *Modeling of Complex Systems*, Academic Press, New York, 1978.

[8] B.P. Zeigler, "Hierarchical Modular Modeling/Knowledge Representation," *Proc. 1986 Winter Simulation Conf.*, pp. 120-137, 1986.

[9] A.F. Sisti, "A Model Integration Approach to Electronic Combat Effectiveness Evaluation," *RL-TR-89-183*, October 1989.

[10] A. W. Naylor and G. R. Sell, *Linear Operator Theory in Engineering and Science*, Springer Verlag, New York, 1982.

[11] H.L. Royden, *Real Analysis*, Macmillan Publishing Company, Ney York, 1988.

[12] G. Franklin, J.D. Powell, and A. Emami-Naeni, *Feedback Control of Dynamic Systems*, Addison-Wesley, New York, 1991.

[13] K. Ogata, *Discrete Time Control Systems*, Prentice Hall, New Jersey, 1987.

[14] T. Kailath, *Linear Systems*, Prentice-Hall, New Jersey, 1987.

[15] H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*, Wiley-Interscience, New York, 1972.

[16] L. Ljung, *System Identification: Theory for the User*, Prentice-Hall, New Jersey, 1987.

[17] W. H. Press S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in FORTRAN*, Cambridge University Press, New York, 1986.

[18] A. C. Antoulas and J. C. Willems, "A Behavioral Approach to Linear Exact Modeling," *IEEE Trans. on Automat. Contr.*, Vol. 38, no. 12, December 1993.

[19] J. P. Norton, *An Introduction to and Identification*, Academic Press, New York, 1988.

[20] N. K. Sinha, and B. Kusta, *Modeling and Identification of Dynamic Systems*, Van Nostrand Reinhold, New York, 1983.

[21] M. Kuijper and J. M. Schumacher, "Input-Output Structure of Linear Differential / Algebraic Systems" *IEEE Trans. on Automat. Contr.*, vol. 38, no. 3, pp. 404-414, March 1993.

[22] D. C Montgomery, *Design and Analysis of Experiments*, John Wiley & Sons, New York, 1991.

[23] D. Belsley, E. Kuh, R. Welsch, *Regression Diagnostics*, John Wiley & Sons, New York, 1980.

[24] D. Caughlin, *Final Report, Modeling Techniques and Applications, Volume I.* USAF Contract F30602-94-0110, Rome Laboratory/IRAE, 32 Hangar Rd, Griffis AFB, NY 13441-4114, December 1995.

[25] L.B. Anderson, et al, "SIMTAX, A Taxonomy for Warfare Simulation," Workshop report taken from the *Catalog of Wargaming and Military Simulation Models, 11th Edition*, Force Structure, Resource, and Assignment Directorate (J-8), The Joint Staff, Washington, DC 20318-8000, September 1989.

[26] J. H. Blakelock, *Automatic Control of Aircraft and Missiles, 2nd Edition*, Wiley-Interscience, New York, 1991.

[27] L. Ljung, *System Identification Toolbox for use with MATLAB*, The MathWorks, South Natic, Mass, 1991.

[28] J. M. Maciejowski, *Multivariable Feedback Design*, Addison-Wesley, New York, 1989.

[29] P. S. Maybeck, *Stochastic Models, Estimation, and Control, Vol 2*, Academic Press, New York, 1982.

[30] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, New York, 1965.

[31] L. Ingber, "Statistical Mechanics of Combat and Extensions," reprint from *Toward a Science of Command, Control, and Communications*, AIAA, December 1993.

[32] B. D. O. Anderson, and J. B. Moore, *Optimal Filtering*, Prentice-Hall, New Jersey, 1979.

[33] A. Gelb, *Applied Optimal Estimation*, The M.I.T. Press, Cambridge. John Wiley & Sons, New York, 1974.

[34] D. Caughlin, "New Procedures to Metamodel Simulations," *Proceedings of the 6th Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, March 1996.

[35] G. Franklin, J.D. Powell, and M. L. Workman, *Digital Control of Dynamic Systems*, Addison-Wesley, New York, 1990.

[36] J. Juang, et.al. "Identification of Observer/Kalman Filter Markov Parameters: Theory and Experiments," pp 320-329, *Journal of Guidance*, Vol 16, No. 2, 1993.

[37] R. E. Maine, and K.W. Iliff, "Formulation and Implementation of a Practical Algorithm for Parameter Estimation with Process and Measurement Noise," *SIAM Journal of Applied Mathematics*, Vol. 41, No. 3, 1981.

[38] Goodwin, Payne, *Dynamic System Identification.* Academic Press, New York, 1977.

[39] J. E. Murray and R. E. Maine, *pEst Version 2.1 Users Manual*, NASA Technical Memorandum 88280, Dryden Flight Research Facility, Edwards, CA, September 1987.

[40] A. L. Ingber, "Simulated Annealing: Practice versus Theory," Reprint from *Journal Mathl. Comput. Modeling.* December 1993.

[41] A. L. Ingber, "Draft of Statistical Mechanical Aids to Calculating Term Structure Models." *Journal Phys. Rev*, Vol 42, 1990

[42] *A Practical Guide to Splines*, Applied Math Sciences, Vol 27, Springer Verlag, New York, 1978.

[43] W. E. Larimore, "System Identification and Filtering of Nonlinear Controller Markov Processes by Canonical Variate Analysis," Final Report for AF Office of Scientific Research, October 27, 1989.

[44] W. E. Larimore and J. Baillieul, "Identification and Filtering of Nonlinear Systems Using Canonical Variate Analysis," pp 635-640. *Proceedings of the* 29^{th} *Conference on Decision and Control*, 1990.

[45] L. Breiman, and J.H. Friedman. "Estimating Optimal Transformations for Multiple Regression and Correlation," pp 580-598, *Journal of the American Statistical Association*, Vol. 80, No. 391, September, 1985.

AUTHOR BIOGRAPHY

DON CAUGHLIN is Acting Director of the Space and Flight Systems Laboratory at The University of Colorado at Colorado Springs. He received a BS in Physics from the Air Force Academy, an MBA from the University of Utah and MS and Ph.D. degrees in Electrical Engineering from the University of Florida. His research interests include modeling and simulation, system identification, pattern recognition and intelligent control. Dr. Caughlin has over 28 years experience as an experimental test pilot, chief scientist, research scientist, program manager and was also Associate Dean of the School of Engineering at the Air Force Institute of Technology. He is a senior member of IEEE and AIAA and a member of the Society of Experimental Test Pilots.

APPENDIX

Notation

$B \subseteq U$	B is a subset of T and B may equal T
$\overline{B^{pc}}$	Closure of B
$Diag\{\gamma_1 \geq \gamma_2 \geq, \ldots, 0\}$	Diagonal matrix with elements $\{\gamma_1 \geq \gamma_2 \geq \ldots., 0\}$
$E\{\cdot\}$	Expected value
$R(\sigma^L, \sigma^{-l})$	Representation of a polynomial operator in the shift. Also may use dummy variables P, Q, and M
$R^{g \times q_1}$	Polynomial matrix of dimension g, q_1
q^{-1}	Backward shift operator
$f_1, f_2 : U \rightarrow E$	The functions f_1, f_2 transform or map U into E
$\{U \mid L\}$	The set U defined by the property L
$cov\cdot$	Covariance
:	"such that"
$\times$	Cartesian product
$\equiv$	Congruence
$\in$	Is in, belongs to
$\exists$	There exists
$\Leftrightarrow$	If and only if
$\forall$	For all
$\frac{\partial[]}{\partial x}$	Partial derivative
$\sup_{p,f}$	Supremum
$\Pi_{j=1}^{i}$	Product of terms from $j-1$ to i
$\|\|\|\|$	Frobenius norm of a matrix
$Pr\{a \leq x_i \leq b\}$	Probability that x_i is grater than a and less than b

Symbols

$A(q)$	Polynomial that multiplies the output variable
$A(\theta)$	Discrete time state transition matrix
$B(\theta)$	Discrete time input matrix
B	Behaviors – outcomes recognized by the model
B_l	Behavior allowed by the inclusion of latent variables
B_s	Behavior that satisfies the axiom of state
$B_{[t_1,t_2]}$	Behavior over the closed interval $[t_1, t_2]$
$B(q)$	Numerator polynomial that multiplies the input variable
$B(t)$	Stochastic input matrix
$C(\theta)$	Discrete time output matrix
$C(q)$	Numerator polynomial that multiplies the error term
D_m	The values over which θ may range in the metamodel set $M(\theta)$
D	The data set used to generate the metamodel
$D(\theta)$	Discrete time feedthrough matrix
$D(q)$	Denominator polynomial that multiplies the error term
E	Abstract set
$e(t)$	Time varying error
F	A Field - a set that satisfies certain algebraic and order properties
$F(q)$	Denominator polynomial that multiplies the input variable
$F(\theta)$	Continuous time state transition matrix
$f(\cdot)$	Nonlinear state propagation function
$G(q)$	Transfer function between the input and output
$G(t)$	Stochastic error gain matrix
$G(\theta)$	Continuous time input matrix

Symbols (cont.)

$H(q)$	Transfer function between the error term and the output
$H(\theta)$	Continuous time output matrix
$h(\cdot)$	Nonlinear output function
I/O	Input – Output
$J(\hat{\theta})$	Cost (objective) function based on θ
$K(\theta)$	Discrete time error gain matrix
$K_u(q, \hat{\theta})$	A matrix filter of dimension $d \times m$
L	Set of latent variables
L^q	Linear shift invariant space
$L(q)$	Stable linear filter
$L(\theta)$	Continuous time error gain matrix
$l()$	Norm for the prediction error used in the Prediction Error Criterion
$\mathcal{M}$	Metamodel structure (mapping)
M	Class of models, model set
$M(\theta)$	Set of metamodels that results from the metamodel structure, For a discrete state space equation, this term multiplies the discrete process noise
M	Model from the set of metamodels
MSE	Mean square error matrix
$\dot{\hat{m}}$	Derivative of the estimate of the stochastic mean
P	Parameter space, A set of parameters
P	State space error covariance
$P(\theta)$	Probability densities
$Q(t)$	Continuous process noise correlation kernel
Q_d	Discrete process noise correlation kernel
R	Set of real numbers
R^d	Euclidean d-dimensional space
R_d	Discrete measurement noise correlation kernel
$R(t_i)$	Sampled data measurement noise correlation kernel
S	System map
S_d	Discrete cross correlation kernel
$S(t)$	Continuous cross correlation kernel
T	The time axis
(t_i^-)	Time extrapolated to the next time step
(t_i^+)	Time after a measurement is taken
U	Universal set of outcomes produced by a phenomenon
u	Input space/variable, (t) indicates time varying
u_a	Actual control inputs
(U, B)	Elements of the model class
W^T	The set of all maps from T to W
W	Signal space, Weight for an error criterion
w	Member of the signal space
$w(t)$	Additive white process noise
$w_d(t_i)$	Discrete additive white process noise

Symbols (cont.)

X	State Variable vector
x	State Variable
$\hat{x}$	State Variable estimate
$\dot{x}$	Derivative of the state variable
x_a	Actual states
$\dot{x}_a$	Actual state derivatives
y	Output space/variable, (t) indicates time varying
$\hat{y}$	Estimate/prediction of the output variable
y_a	Actual output
y_m	Model output
Z^N	Set of observation (measurement) vectors of length N
(Z, R^q, B)	Linear time-invariant dynamical system where $T = Z_+$ and $W = R^q$
$z(t_i)$	Sampled data observation
$d\beta(t)$	Brownian motion of diffusion strength $Q(t)$
Δ	Finite memory span
Δy	Finite change in y
ε	Prediction error
ε_F	Filtered prediction error
π	Map from the parameter set to the model set
$\Phi(t, t_0)$	Continuous time state transition matrix
$\phi(t_i)$	Input (possibly nonlinear) data (functions)
$\phi^T(t)$	Transpose of the vector of regressors
Σ	Dynamical system, covariance of θ in MAP estimation
Σ_l	Dynamical system with latent variables
σ^t	Differentiation or the time-shift operator discrete time systems
σ	Standard deviation
θ	The vector used to parameterize the model
θ_*	Value of the "true" parameter vector
$\hat{\theta}$	Estimate of the parameter vector
θ^T	Transpose of the parameter vector (note: all $(\cdot)^T$ are the transpose except W^T
$\nu(t_i)$	Additive white (sampled data) measurement noise
$\wedge$	Concatenation

A Summary of Model Abstraction Techniques

A. F. Sisti[a] and D. Caughlin[b]

[a] Rome Laboratory / IRAE
32 Hangar Rd
Rome NY 13441-4114 USA
sistia@rl.af.mil

[b] Space and Flight Systems Laboratory
University of Colorado at Colorado Springs
Colorado Springs, Colorado 09033-7150 USA
donc@mozart.uccs.edu

ABSTRACT

This paper presents an overview of model abstraction methods. Model abstraction methods are techniques that derive simpler conceptual models while maintaining the validity of the simulation results. These methods include variable resolution modeling, combined modeling, multimodeling, and metamodeling. In addition, some taxonomies include approximation, aggregation, linear function interpolation, and look up tables as model abstraction methods. We discuss these methods in a general framework to assist in understanding the applicability of the various model abstraction methods.

Keywords: Simulation, Metamodel, Model, Abstraction

1. INTRODUCTION

A model is a structure that can be used for understanding the behavior of a system.[1] The development of the model then is an abstraction of a "real world" concept or system where we have analyzed the "real world" system, determined the behaviors that will be addressed by the model and determined a structure for its representation. The model can be a physical structure such as a wind tunnel model used to determine the aerodynamics of an aircraft, or it could be a conceptual model represented by interactions, a system of equations, or a simulation. We will restrict our attention to simulation models.

In a slight variation from Reference [2], we define a model abstraction technique as a method (simplifying transformation) that derives a simpler conceptual model from a more complex model while maintaining the validity of the simulation results with respect to the behaviors addressed by the simpler model.

There are two general types of model abstraction techniques: these are the "Direct" and "Inverse" methods.

First, a more abstract model could be developed by applying basic principles to generate a more abstract (approximate) version of the "real world" system. This would be an example of direct modeling. Direct modeling is characterized by a specification of the elements of the model. Complicated systems are modeled by "tearing" a system into its components, modeling these components in a process called "zooming," and then interconnecting these components to construct a "physical" realization of the system.[3–5] The level of abstraction is controlled by the detail of the specification. The model reveals the structure of the theory and allows the prediction of the response to exogenous inputs as a function of the state of the system. The solution of this modeling problem requires an understanding of the process being modeled and methods to express this understanding. With the exception of the metamodeling technique presented in Section 2.6, all of the abstraction techniques discussed are direct methods

Inverse modeling begins with the input-output data generated by the "real world" system or the high fidelity model or simulation and develops the abstract model from the data. In this case, we have some estimate (measure) of the input and output response but do not have a complete characterization of the process by which the outputs are generated. System identification methods are used to generate a mathematical approximation between the inputs and responses.

This paper presents an overview of both direct and inverse model abstraction methods.

The literature identifies variable resolution modeling, combined modeling, multimodeling, and metamodeling as specific abstraction methods. In addition, some taxonomies include approximation, aggregation, linear function interpolation, and look up tables as model abstraction methods. In Section 2, we discuss abstraction methods from a more general perspective that follows the historical development of model abstraction methods. Section 2.1 introduces the subject with Model Based Abstraction Techniques. These methods eventually developed into the Discrete Event formalism which is discussed in Section 2.2. Section 2.3 moves from the Model Based approach to Process Based Abstraction techniques. This approach has developed into Multimodeling which is discussed in Section 2.4. Section 2.5 discussed Qualitative Based techniques, while Section 2.6 discusses the only inverse method presented - Metamodeling. Section 3 begins with a model Abstraction Taxonomy discussed in [2] and concludes with a proposed new taxonomy. Section 4 concludes the paper.

2. TAXONOMIES OF ABSTRACTION TECHNIQUES

One of the problems in discussing model abstraction techniques and applications is a lack of uniform terminology. Closer inspection of the literature, however, indicates that many of the concepts are the same and that the differences arise from the perspective of, or terminology used by, the author.

Since a discussion of abstraction techniques is a function of perspective, we begin with a discussion of abstraction techniques provided by the general approach of the author. This discussion leads to several taxonomies of abstraction techniques which we will attempt to reconcile.

2.1. Model Based Abstraction Techniques

Zeigler's development of the Discrete Event System (DEVS) Formalism was based on a model based simulation architecture where the model is described by a formal object called a system specification. The elements of the modeling approach included the real system, experimental frame, base model, lumped model, and the computer.

Zeigler used the term "base model" to express the most detailed model and the term "lumped model" as the simpler (abstracted) model.[6] Being hierarchical in nature, this architecture naturally led to model abstractions and abstraction techniques.

As stated, the description of the object was through the system specification which involved an abstraction from the base to the lumped model. The system specification was cast into a hierarchy of levels:

1. I/O relation observation
2. I/O function observation
3. I/O system
4. Iterative specification
5. Structured system specification
6. Network of specifications

with a corresponding hierarchy of preservation relationships (morphisms) that preserve the validity of the model.

The transition from the base mode to the lumped model involved simplification (abstraction). Initially, he discussed four general categories of abstraction techniques:

1. **Dropping Components**. Since all factors are not equally important, the first technique is to ingore components, descriptive variables, or interaction rules. This is similar to an engineering approximation and is a reduction in the complexity of a model by eliminating factors which least affect the response of interest.

2. **Stochastic approximation**. This abstraction technique replaces deterministic variables by random variables. This technique reduces fidelity by representing a higher order deterministic process by possible outcomes that are selected based on some probability distribution.

3. **Coarsening the Ranges of Descriptive Variables**. This technique can be as simple as a straight reduction in variable range or considering a reduced set of allowable values for the variables that results from a many-to-one mapping of the variable intervals. This reduction can also be implemented by a reclassification of the variable attribute, such as the replacement of a string variable with a Boolean variable indicating the attribute as empty or non-empty.

4. **Grouping**. The final category is grouping components and aggregating their variables. It can be considered as compounding and then coarsening the resultant compound range.

Along with the simplification or abstraction we must consider the validity of the simpler model. The measure here is a homomorphism or similar structure, and elements of the measure are the preservation of the time advance mechanism, preservation of transition functions, and preservation of output functions.

This early structure evolved into DEVS which explicitly includes the level of model abstraction as a parameter in the definition of the model.

2.2. Discrete Event Based Abstraction Techniques

The DEVS Formalism[7,8] provides a systems theoretic basis for modeling and simulation that specifies the system in terms of the level of abstraction and morphisms. DEVS is based on the relationship between the system, the model, and the implementation of the model. Since this relationship inherently includes the level of fidelity (abstraction), model components are organized by their level of abstraction and the degree in which they preserve the relationships of the model specification. This organization is contained in a framework for structuring the model – the System Entity Structure.

The System Entity Structure (SES) supports a hierarchy of abstraction levels and is based on modularity and coupling. Modularity describes the model with its inputs and outputs through which all interaction with the external world is mediated. Coupling describes the interconnections of the input and output ports of simpler models into more complex models.

The SES directs the synthesis of models and combines the decomposition, taxonomic, and coupling relationships. In this formalism, a system has a time base, inputs and outputs, states, and functions that define the relationships between the system being modeled and the simulation that will represent that system. As such it is a structured abstraction technique that maps the system into atomic (basic) and coupled (multi-component) models.

Within DEVS there are functions that operate on hierarchical model structures (deep-devs, flat-devs, flat-all, inverse-transform, etc.). Consequently, the process of developing a DEVS representation explicitly addresses the question of deriving more abstract models from the specification of the system and as such should be considered as an abstraction technique.

2.3. Process Based Abstraction Techniques

Fishwick defines an abstraction network as an ordered pair of models and abstraction relationships.[9] The ordered set of models represents the inputs, outputs, and relationships of the underlying system at different levels of abstraction. The abstraction relationships are based on the processes represented by a model. He describes seven forms of abstraction:

1. **Abstraction by Representation**. This form uses a different representation to present the same information

2. **Induction**. Induction aggregates several behaviors into a single representation.

3. **Reduction**. Reduction based on ????

4. **Total Morphism**. Morphisms define mappings between two models. Total morphisms are relationships that accomplish a complete mapping of the system that preserves the features of the system.

5. **Partial Morphism**. A morphism that does not necessarily preserve all of the relationships and functions during the mapping.

6. **Sensory Abstraction**. Is the generation of behavior that visually represents the behavior of the system being modeled.

7. **Cerebral Abstraction**. This is the highest level of model abstraction based on intuition.

2.4. Multimodeling Based Abstraction

Fishwick and Lee have continued to develop process based abstraction techniques and have proposed that variable resolution modeling, combined modeling, multimodeling and metamodeling can be categorized as either **behavioral** or **structural** approaches.[10]

Variable structure models include in the model description sufficient information that allows the model to change its own structure. Multimodels contain structurally compatible component models with distinct behaviors that play a mutually exclusive role. While multimodeling provides for a hierarchical structure, selection of components in each level is dependent on the lower level component selection. In each of these model abstraction methods, either the behavior or the structure is addressed.

Behavioral abstraction addresses the level of complexity by approximating the behavior of the system. This category of abstraction simplifies a component by replacing it with something more generic but produces similar behavior. In some respects, behavioral abstraction is equivalent to multimodeling but the components are black boxes defined only by the input-output map.

Structural abstraction defines the levels of abstraction and chooses which model types to use at each level. Structural abstraction focuses on the structure of the model and not necessarily the resulting behavior. Structural abstraction isolates the abstraction levels so that each level can be executed independently from the other levels without requiring knowledge of the detailed internal structure of the other levels. Within structural abstraction we can address either the values obtained by the model (data) or the model itself.

Data abstraction compresses information obtained from the model. For example, with data abstraction we could represent a trajectory by its symbolic value, mean, variance, interval, or fuzzy set. Model abstraction addresses the strucure of the model itself and can be homogeneous or heterogeneous. Homogeneous model abstraction restricts the structural abstraction to a single representation - conceptual, declarative, functional, constraint, and spatial. Heterogeneous abstraction allows multiple representations under one structure.

The taxonomy that results from this approach is depicted in Figure 1.

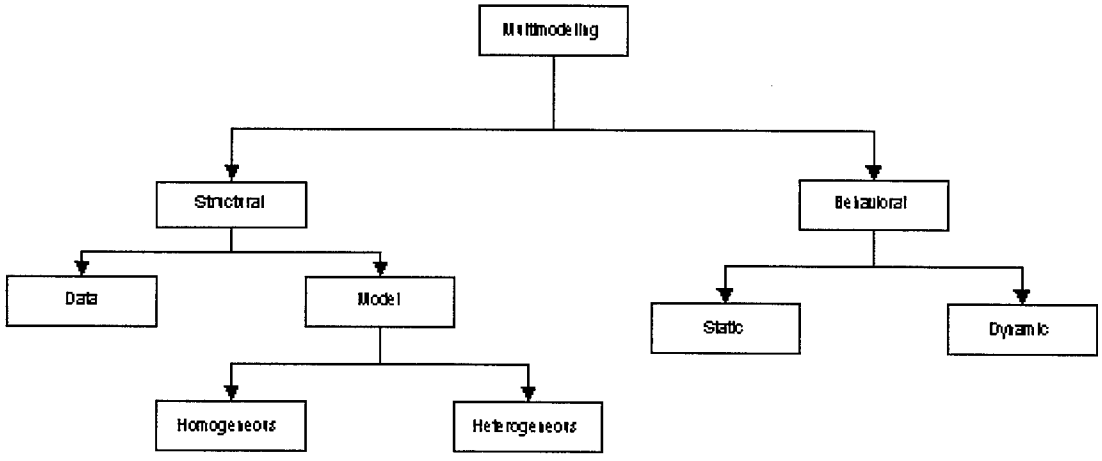


Figure 1. Multimodel Taxonomy.

2.5. Qualitative Simulation Based Abstraction Techniques

Our next approach is based on a qualitative approach to modeling. Qualitative simulation is based on qualitative differential equations and qualitative processes which describe the relationships that represent the system. These relationships include a set of variables, a quantity space, a set of variable constraints, and a set of transitions. Qualitative simulation is distinguished by the fact that the quantity space of the variable is not the domain of numbers but "landmark" values that have specific meaning (e.g. hot, cold, empty, full, etc.).

Abstraction techniques identified in the qualitative simulations literature include;

1. **Behavior Aggregation**. Characterizes all possible behaviors at different levels of detail while eliminating irrelevant distinctions.
2. **Structural Abstraction**. Aggregates components that are close.
3. **Functional Abstraction**. Aggregates components that are elements of the same function.
4. **Chatter Box Abstraction**. A chatter box is a state space region where qualitative derivatives are allowed to vary while the qualitative values of other variables remain the same. Chatter Box Abstraction eliminates chatter by reducing the region to a single state.
5. **Model Decomposition**. Divides the model into loosely connected components that are modeled separately while specifically addressing component interactions.
6. **Time Scale Abstraction**. There are two aspects to Time Scale Abstraction. One aspect aggregates behavior over an interval and represents that behavior at a particular point (start, mid, or end) on the interval. The second aspect of Time Scale Abstraction is a particular form of model decomposition where the decomposition of complex inter-related systems is focused on parts of the system that operate at different time scales. This technique separates the time scales so that to the fast system, the slow system is constant; and to the slow system, the fact system is instantaneous. These approximations allow a corresponding reduction in the complexity of the interactions between the two systems.
7. **Quantitative Abstraction**. Ignores small differences in the value of variables.

2.6. Metamodeling

All of the abstraction methods presented to this point addressed very general modeling issues. Our final abstraction methodology does not address the entire model space but provides a specific set of methods to support some of the above methods.

By definition, a metamodel is a model of a model. We restrict the use of the term metamodel, however, to mathematical approximations of the system relationships defined by a high fidelity model or simulation.

As an abstraction, a metamodel is a projection of the model onto a subspace defined by new constraints or regions of interest. Abstract models developed using the direct techniques presented above are "stand alone" versions. The relationship between the real system, a high fidelity model, and the more abstract model is contained in the two mappings from the underlying system to each of the models. Also, these techniques require an *a priori* understanding of the structure of the elements and the interconnections between these elements at the specific level of fidelity selected.

The metamodeling technique we present here is based on a solution of the inverse problem and is shown in Figure 2. This technique is a structured metamodeling method that simplifies the metamodeling process to two phases: problem definition and an iterative metamodeling process.

In the problem definition, we begin with an analysis of the metamodel requirements and the simulation under study. We then progress to the description of the system (not the model) so that we will be able to select a metamodel structure that matches both the requirements and simulation that we are going to metamodel. We determine the purpose of the metamodel. In the definition of this purpose, we have identified the input and response that we are interested in and determined the important characteristics of these data. Also for this purpose, we have defined the region of interest, selected validity measures, and specified the required validity.

In addition to the purpose of the metamodel, we also characterize the simulation that were are trying to model. In our approach, we do not address the representation of the metamodel or assumptions that will be made in its realization. However, data generated by this step provides a clear statement of the metamodel purpose and the characteristics of the simulation that will be modeled. As will be seen in the next section, this data directly matches the decisions that must be made in the selection of the model set.

2.6.1. Structured Metamodeling Method

Prevalent metamodeling approaches required too many decisions involving: *a priori* knowledge; the data; possible metamodel sets; and rules to determine the best model set to realize the data. Each decision was a complex function of *a priori* information and prior selections in the metamodeling process.

In reality, all "real world" systems are complex, large scale interconnections of continuous-discrete, nonlinear, infinite-dimensional components. We will approximate these systems with lumped parameter, parametric, finite dimensional models that can be grouped into sets.

A new taxonomy of metamodel sets and identification methods was developed that allows the separation of the metamodeling process into a set of sequential decisions based on *a priori* information. This decision sequence is shown in Figure 3. The purpose of the procedure is to match the problem definition and characterization of the simulation to the behavior allowed by the metamodel set.

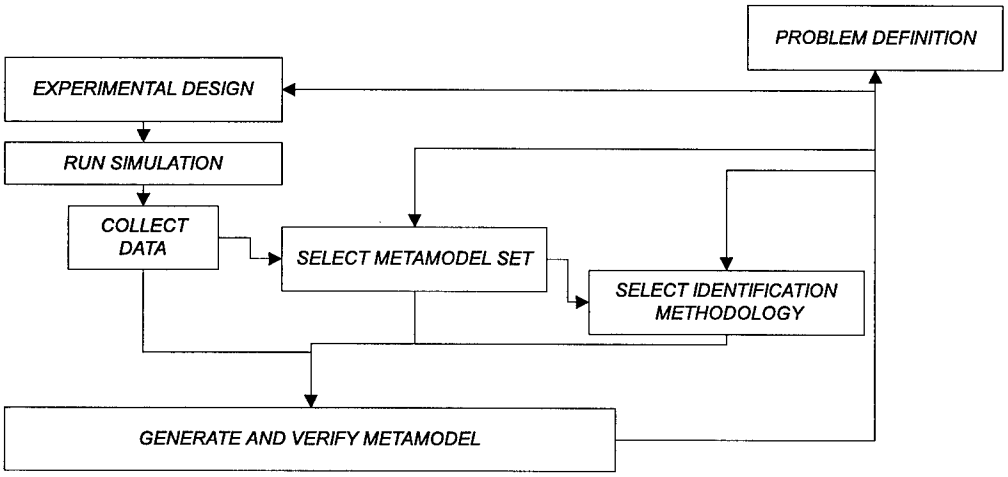


Figure 2. Iterative Metamodeling Process

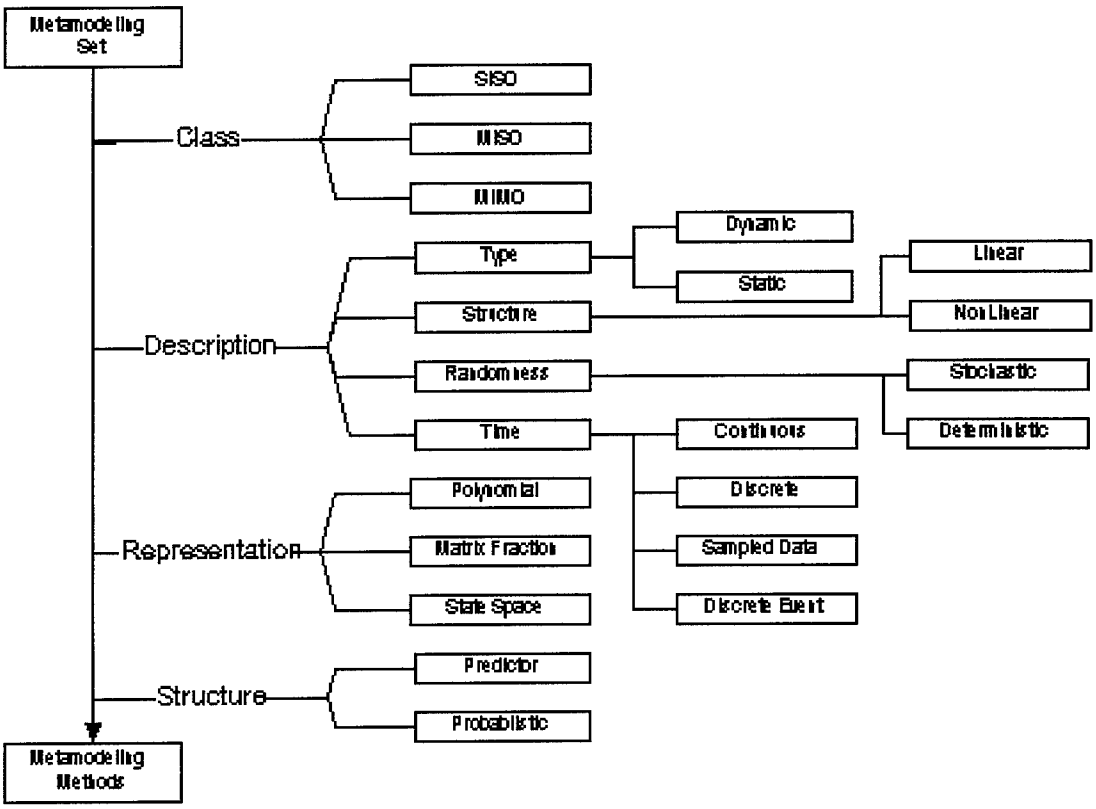


Figure 3. Metamodeling Set Determination

As seen from Figure 3, selection of the metamodel set is clearly defined by the system description, system class, representation, and metamodel structure. Data for all of these selections come directly from the problem definition step.

In each of these model sets, a most powerful unfalsified model will exist (given that the certain requirements are met).[3] Consequently, the performance of the metamodel will be limited by the match between the metamodel set and the actual system that generated the behavior.

2.6.2. Generation of the Metamodel

We have defined the problem and selected a metamodel set that matches this definition. Now we must select a method to generate the model parameters.

There are many taxonomies used in the literature to categorize identification methods. Methods can be referred to as off-line or on-line. Also, they can be classified as either open-loop or closed-loop methods. Further classification can be made as nonparametric, frequency domain, and as parameter identification methods. The number of methods and classification schemes complicated method selection, and none of the classifications really addressed the application of a method based on the metamodel set.

By selecting the metamodel set as shown in Figure 3, a new structured metamodeling method was possible that addresses this complexity. The structure is based on the fact that the construction of a metamodel (selection of the parameters used for the projection) is determined by the metamodel set, method of identification, and identification criteria.

Analyzing the method of identification and identification criteria,we can reduced the parameters identification methods to four approaches shown in Figure 4. A summary discussion of these elements is included in Reference [15], additional details are found in [11].

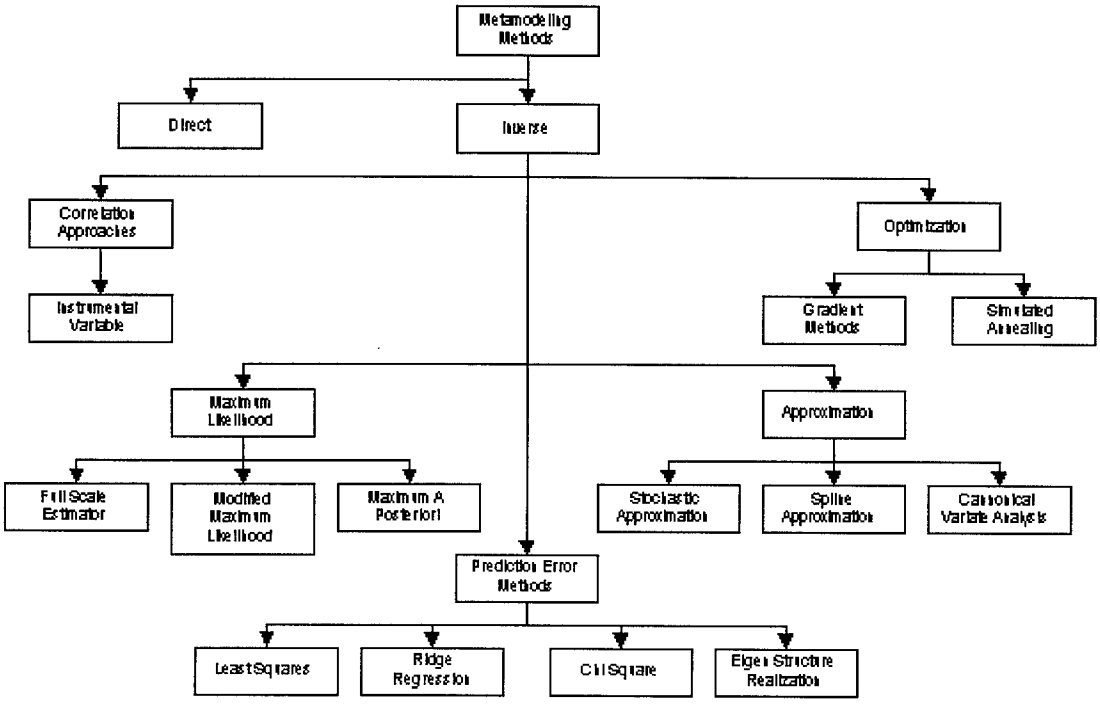


Figure 4. Taxonomy of Metamodel Methods

3. MODEL ABSTRACTION TAXONOMY

3.1. Initial Model Abstraction Taxonomy

An analysis of the above techniques was accomplished in [2], and is presented in Figure 5.

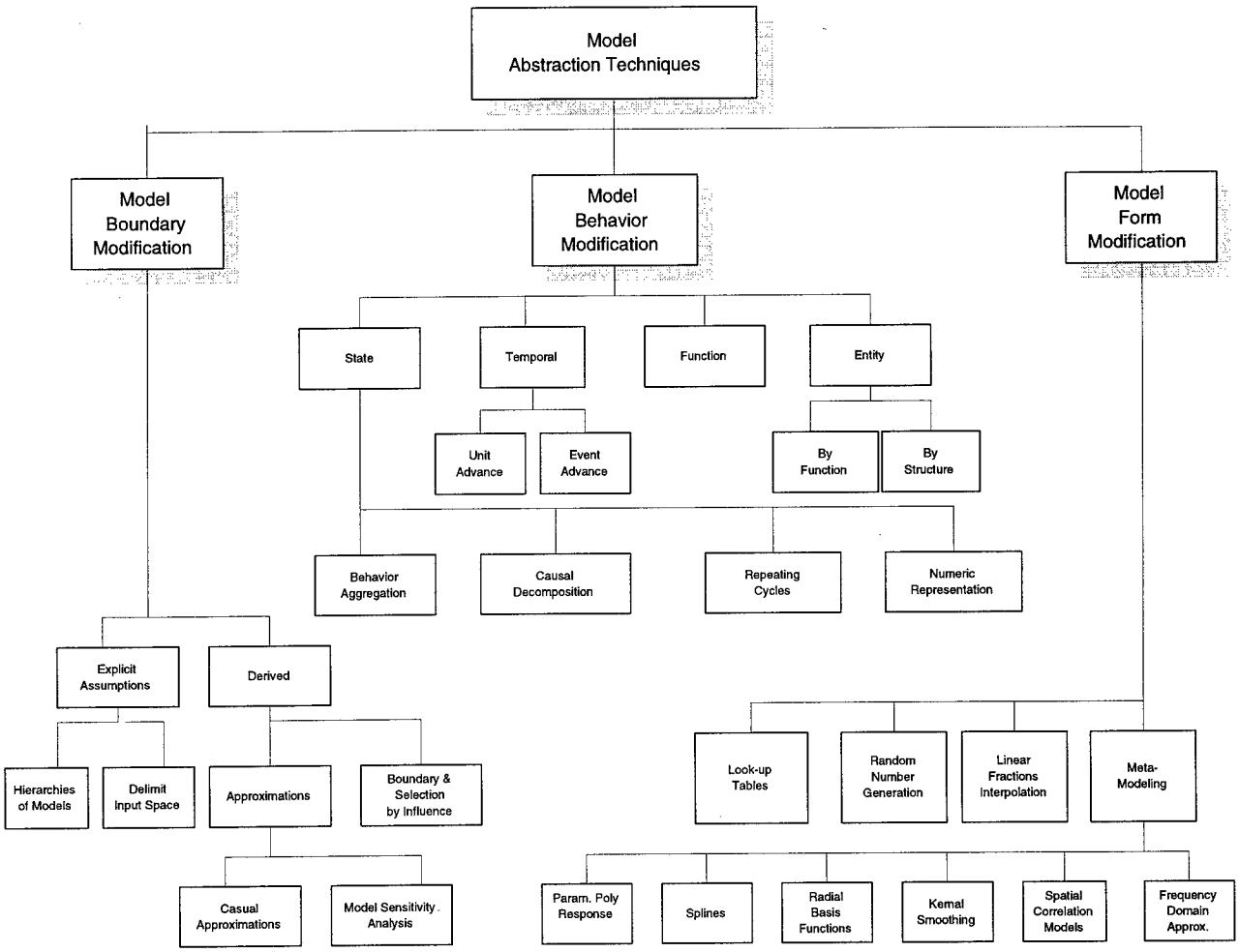


Figure 5. Model Abstraction Taxonomy.

While complete, some simplifications can be made so that the distinctions in types of abstraction come through more clearly. This taxonomy attempts to display all of the possible abstraction methods and their resulting representations in a single flowchart. This construct requires that we consider both the underlying process and how the process is accomplished. If carried to completion, the resulting two dimensional diagram is excessively complex and loses it's ability to convey any information.

3.2. Proposed New Model Abstraction Taxonomy

Since the process of abstracting a model usually entails multiple passes operating on the same or derived models, we will approach a taxonomy of abstraction techniques in the same manner. The methods in the taxonomy we describe are applied iteratively until the desired representation and level of abstraction is obtained.

The taxonomy addresses processes that are available for model abstraction and, with the exception of separating the model and the data, does not address the component of the model that the method operates on. In the structural

techniques we will delineate between the structure of the model and the structure of the data. We divide the problem space into models and data because the model is the structure that can be used for understanding the behavior, the data is the interaction of the model with its environment. Structural abstractions can be applied to either space.

The focus on the process means that we do not consider the result of the process as part of the taxonomy. For example, application of structural abstraction applied to the model (as opposed to the data) can result in different structural classes that can be used in the same overall model. In Figure 1 above, this results in the Hetrogeneous Multimodel.

Some portions of the taxonomy, however, are based on the initial condition of the process. Direct behavioral techniques can be thought of as Object-Oriented (OO) or Structured (functional). This distinction is similar to the distinction between OO Modeling and Design and Systems Engineering. In the OO approaches we analyze the model from the bottom up looking at the entities that are used to describe the system and their interaction with the environment. In the Structured Approaches we begin with a top down analysis and decompose or abstract the system beginning at the highest level.

With these few introductory remarks, Figure 6 presents this new taxonomy of Model Abstraction Techniques.

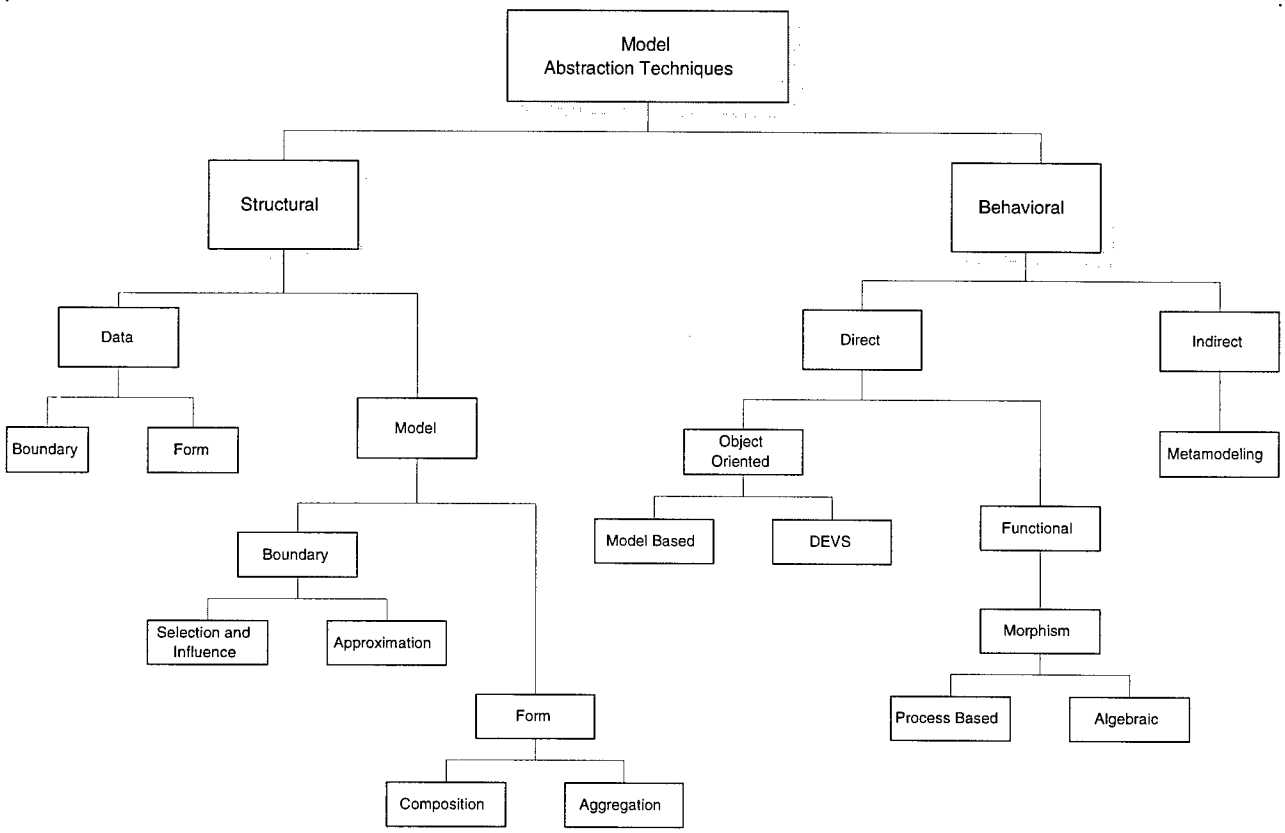


Figure 6. Proposed Model Abstraction Taxonomy.

Unfortunately, the length of the paper prevents a complete description of each branch or the specific mention of every technique in the branches. We do however, provide a general discussion of the major aspects. The Taxonomy was slightly simplified for presentation. The presentation of Structural Techniques that operate on data do not continue below "Boundary" and "Form" but the remaining options are identical to Structured Model abstraction techniques. The inverse metamodeling techniques shown in Figure 4 were not repeated and should be appended to Figure 6 in the appropriate location.

The primary boundary is between "Structural" and "Behavioral" abstraction. In behavioral abstraction we address the behaviors that we are going to model and how these behaviors are related to the underlying model that we are abstracting. We can accomplish this by working directly with the behavior from a Structured or Object Oriented perspective or we can abstract the behavior indirectly through the data using Metamodeling.

The OO methods are divided into "Model Based" and "DEVS". Although DEVS is an outgrowth of a model based approach, this distinction was maintained because of the additional constraints placed on DEVS when compared to the model based approach. An example of an additional constraints is the restriction to a discrete event system. It is possible to use model based approaches for continuous, discrete or sampled-data models.

Under Direct Structured Behavioral approaches we divide morphisms into those that are purely mathematical in nature and those that would allow other types of abstractions that are based on process relationships.

Structural Model abstraction techniques can be applied to the data or the form of the model. If we address the form of the model we can approach the abstraction through composition or aggregation. Both methodologies can be applied to portions of the model or to the model as a whole. Aggregation is a decomposition of the model into parts where the sum of all of the parts now represent the model. In composition we do not decompose the model into mutually exclusive elements but into components that are part of the original model. These components represent characteristics of the model but there is no assumption that the sum of the components represent the entire (abstracted) model.

Structural Data abstraction maintains the behavior and the structure of the model but allows the analyst to address the representation of the input or output. Here we could group variables, modify intervals, etc. as long as the full behavior is not modified in any manner.

If we address the boundary of the model we are changing the domain and/or range of the model. We can make this change by direct selection or by approximating the boundary of the model in a different representation.

4. CONCLUSION

We have reviewed various model abstraction techniques and suggested a taxonomy that includes and unifies those techniques. This new taxonomy was presented in Figure 6 and is based on the following constructs:

1. The taxonomy addresses the processes that are available for model abstraction and does not address the component of the model that the process operates on.
2. The taxonomy does not consider the result of the process.
3. Taxonomy we describe is applied iteratively until the desired level of abstraction is obtained.

Because the taxonomy does not consider the model components or the result of the abstraction process, it appears to be quite different from those proposed in the past. It is not possible to use this taxonomy to determine the type of the resulting (abstract) model.

REFERENCES

1. V. Vemuri, *Modeling of Complex Systems*, Academic Press 1978.
2. Fredrick K. Frantz and A. John Ellor, "Model Abstraction Techniques," Computer Sciences Corporation , Integrated Systems Division, One MONY Plaza, Syracuse, New York 13202, 18 August 1995.
3. J. C. Willems, "Paradigms and Puzzles in the Theory of Dynamical Systems," *IEEE Trans. on Automat. Contr.*, vol. 36, no. 3, pp. 259-294, March 1991.

4. B.P. Zeigler, "Hierarchical Modular Modeling/Knowledge Representation," *Proc. 1986 Winter Simulation Conf.*, pp. 120-137, 1986.
5. A.F. Sisti, "A Model Integration Approach to Electronic Combat Effectiveness Evaluation," *RL-TR-89-183*, October 1989.
6. B. Zeigler, *Theory of Modeling and Simulation*, Wiley and Sons, New York, 1976
7. B. Zeigler, *Multifacetted Modeling and Discrete Event Simulation*, Academic Press, San Diego, 1984.
8. B. Zeigler, *Object-Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems*, Academic Press, San Diego, 1990.
9. Paul A. Fishwick and Kangson Lee, "Two Methods for Exploiting Abstraction in Systems," Proc. *AI, Simulation and Planning in High Autonomy Systems*, March 23-27, 1996.
10. Paul Fishwick, "The Role of Process Abstraction in Simulation," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol 18, No. 1, January/Febrauary, 1998.
11. D. Caughlin, *Final Report, Modeling Techniques and Applications, Volume I.* USAF Contract F30602-94-0110, Rome Laboratory/IRAE, 32 Hangar Rd, Griffis AFB, NY 13441-4114, December 1995.
12. D. Caughlin, "A Metamodeling Approach to Model Abstraction," *Proc. 1994 Fourth Annual IEEE Dual Use Technologies and Applications Conference*, May 1994.
13. D. Caughlin, "An Evaluation of Simulated Annealing for Modeling Air Combat Simulations," Proc. 1994 IEEE Dual-Use Technologies and Application Conference, May 1994.
14. D. Caughlin, "Verification, Validation, and Accreditation (VV&A) of Models and Simulations Through Reduced Order Metamodels," *Proc. 1995 Winter Simulation Conference*, December 1995.
15. D. Caughlin, "New Procedures to Metamodel Simulations," *Proceedings of the 6th Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, March 1996.
16. M. A. Zeimer, et. al., "Metamodel Procedures for Air Engagement Simulation Models," *IRAE Technical Report*, Jan 1993.
17. L. Ljung, *System Identification: Theory for the User*, Prentice-Hall, New Jersey, 1987.
18. D. C Montgomery, *Design and Analysis of Experiments*, John Wiley & Sons, New York, 1991.
19. D. Belsley, E. Kuh, R. Welsch, *Regression Diagnostics*, John Wiley & Sons, New York, 1980.
20. L.B. Anderson, et al, "SIMTAX, A Taxonomy for Warfare Simulation," Workshop report taken from the *Catalog of Wargaming and Military Simulation Models, 11th Edition*, Force Structure, Resource, and Assignment Directorate (J-8), The Joint Staff, Washington, DC 20318-8000, September 1989.

A Metamodeling Approach to Model Abstraction

Don Caughlin
Mission Research Corporation, Colorado Springs, Colorado 80919

Abstract— **This paper provides a framework for the application of System Identification techniques to develop suitable Metamodels for tactical simulations used by the Department of Defense. We fill in the framework with concrete definitions and identify specific issues associated with the representation of dynamical systems. Particular attention is given to the discussion of experimental design requirements for Metamodeling Tactical Engagement (usually Discrete Event System - DES) simulations. We demonstrate this approach by outlining the development of a Metamodel for the "Tactical Electronic Reconnaissance Simulation Model."**

1 Introduction

Tactical Simulation models used by the Department of Defense to assess the capabilities of combat systems and tactics are highly complex. It is often difficult to determine the relationship of individual factors to the performance of the modeled process [1]. Consequently, it is not easy to use the results of the model in another simulation or couple multiple models to investigate a larger issue. The result is a proliferation of point designed models and simulations, expensive upgrade and maintenance, and the inability to efficiently answer many of the more difficult questions raised by the acquisition and operational communities [2].

A technique called Metamodeling has the ability to facilitate this type of assessment. As an abstraction, a metamodel is a projection of the model onto a subspace defined by new constraints or regions of interest. Selection of the parameters used for the projection (the construction of a metamodel) involves: *a priori* knowledge; the data; a set of metamodel structures; and rules to determine the best model to realize the data. This paper presents a new paradigm and discusses some of the issues associated with Metamodeling tactical simulations.

The paper is organized as follows: Section 2 introduces Metamodels and techniques available to develop them; Section 3 introduces the Framework that will be used for the Identification of Metamodels from Combat Simulations; Section 4 identifies some special issues associated with the Identification of simulation Metamodels; Section 5 provides an example of using this approach; and Section 6 concludes the paper with results and conclusions.

2 Metamodels

A model is a structure that can be used for understanding the behavior of a system [3]. The model can be a physical structure such as a wind tunnel model used to determine the aerodynamics of an aircraft, or it could be a conceptual model represented by interactions, a system of equations, or a simulation.

Assume that we have a model of a system that cannot be used directly. A solution may not exist, it may be too complicated for a closed-form solution, it may require too much time to numerically determine a particular solution, or it may be a high-fidelity simulation that provides much more detail than we are interested in. Efficient use of this model requires a "black-box" approximation of the causal time dependent behavior of the model – a Metamodel.

A Metamodel is a mathematical approximation of the system relationships defined by another, more detailed model (in our case – a tactical simulation).

2.1 *Metamodeling Techniques*

There are two basic techniques available for Metamodeling: direct and inverse modeling.

First, a metamodel could be developed by applying basic principles to generate a more abstract (approximate) version of the original model. This would be an example of direct modeling. Direct modeling is characterized by a specification of the elements of the model. Complicated systems are modeled by "tearing" a system into its components, modeling these components in a process called "zooming," and then interconnecting these components to construct a "physical" realization of the system [4, 5, 6]. The level of abstraction is controlled by the detail of the specification. The model reveals the structure of the theory and allows the prediction of the response to exogenous inputs as a function of the state of the system. The solution of this modeling problem requires an understanding of the process being modeled and methods to express this understanding.

Metamodels developed using this technique are "standalone" versions. The relationship between the

real system, the original model, and metamodel is contained in the two mappings from the underlying system to each of the models. There is no guarantee that a usable correspondence will exist between the metamodel and the model [7, 8]. Traceability from the high-fidelity model to the more abstract, lower fidelity, metamodel becomes a significant issue. Also, this technique still requires an *a priori* understanding of the structure of the elements and the interconnections between these elements at the specific level of fidelity selected. This, in fact, could be a difficult and risky task and lack of this knowledge is often the reason that a high fidelity simulation was used in the first place.

The second technique develops the metamodel from the Input-Output data generated by the original model or simulation. This technique is an example of the "inverse problem." As difficult as the direct modeling problem may be, the inverse problem is much more complex. In this case, we have some estimate (measure) of the input and output response but do not have a complete characterization of the process by which the outputs are generated. While a properly posed direct problem generally has a solution, the inverse problem usually has multiple solutions out of which an acceptable solution (if it exists) must be selected. This technique explicitly results in a mathematical approximation between the inputs and responses - this is the technique we consider.

It should be noted that there is a significant difference between our approach and much of the prior research. Most of the previous work that could be categorized as Metamodeling consisted of procedures to determine the best polynomial fit to a set of Input-Output data. The researchers concentrated on the statistical properties of the data. In our approach, we are not trying to fit data. We are attempting to identify the underlying processes that define the system that generated the data (or in our terminology - the behavior). The focus is not on statistics but on the system theoretic properties of the manifest behavior.

3 Identification Framework

Given a phenomenon that we would like to describe, we desire a mathematical expression as the model [2] [4]. Assume that this phenomenon produces **outcomes** that are elements of a set U. A model for this phenomenon will probably generate certain of these outcomes and exclude others. Consequently, the outcomes recognized by the model, B, are a subset of the universal set U, and are called the **behavior** of the model. For the inverse modeling problem, we define a model class $\mathcal{M}$ with elements $M = (U, B)$ where $B \subseteq U$ is the behavior of M.

[2] This framework follows the work presented by Willems

Therefore, define a mathematical model as the pair (U, B) with U the universe of outcomes produced by the underlying phenomenon, and B, the behavior of the model. If possible, we can describe the behavior of the model by a set of equations that leads to a behavioral equation representation of the pair (U, B). To accommodate this, consider an abstract set, E, called the equating space, and $f_1, f_2 : U \rightarrow E$. With this space, and the functions f_1, f_2, the behavioral representation for the model becomes (U, E, f_1, f_2).

In summary, the modeling procedure requires that we specify the variables that we want to model (specify the universal set U), and identify the possible outcomes in the behavior. Often, however, we will require additional variables in addition to those we seek to model. These other variables are called latent variables. These variables are required whenever we develop a metamodel by the method of tearing where the system is viewed as the interconnection of subsystems. Consequently, we expand the mathematical model to allow latent variables by defining a triple (U, L, B_f). Here L is the set of latent variables, $B_f \subseteq U \times L$, with $B_f \equiv \{u \in U | \exists l \in L \; such \; that \; (u, l) \in B_f\}$.

A mathematical model is linear if U is a vector space and B is a linear subspace of U. Assume that $U = I \times O$, where I is the input space, O is the output space, and B is the graph of a system map $F : I \times O$ called an I/O map. These assumptions allow an Input-Output model where $(U, B) \Leftrightarrow (I \times O, B) \Leftrightarrow (I, O, F)$. If the past does not contain any information about the future other than the information in the behavioral relationships, the map is nonanticipating. A parametrization of M consists of a set P and a surjective map $\pi : P \rightarrow M$. The set P is the parameter space with $p \in P$ determining the behavioral equations.

3.1 *Dynamical Systems*

Again, the model for a dynamical system is defined in terms of its behavior. A dynamical system is a family of trajectories without reference to I/O maps, variables, or behavioral equations. The system is coupled to its environment and is not defined by a model associated with it. A model for a dynamical system Σ is simply a triple $\Sigma = (T, W, B)$ with $T \subseteq R$ the time axis, W the signal space, and $B \subseteq W^T$ the behavior – the set of all maps from T to W, a family of W-valued time trajectories.

A dynamical system is linear if W is a vector space (over a field F) and B is a linear subspace of W^T. A dynamical system $\Sigma = (T, W, B)$ is said to be time invariant if $\sigma^t B = B$ for all $t \in T$. Where σ^t is the time-shift operator: $(\sigma^t f)(t') \doteq f(t' + t)$.

A dynamical system $\Sigma = (T, W, B)$ is said to be complete if $\{w \in B\} \Leftrightarrow \{w|_{[t_1,t_2]} \in B_{[t_1,t_2]}, \forall t_1, t_2 \in T, t_1 \leq t_2\}$. Completeness is an important property

affecting the mathematical structure that defines the behavioral equations that represent dynamical systems.

Dynamical systems acquire their importance from the fact that they exhibit memory or the potential to model phenomena where the past influences the future. In this context, a dynamical system is said to have a finite memory span $\Delta(\Delta \in T, \Delta > 0)$ if $w_1, w_2 \in B, w_1(t) = w_2(t)$ for $0 \leq t \leq \Delta \Rightarrow \{w_1 \wedge w_2 \in B\}$ [3]. Where

$$(w_1 \wedge w_2)(t) = \begin{cases} w_1(t) & for \quad t < 0 \\ w_2(t) & for \quad t \geq 0 \end{cases} \qquad (1)$$

If $\Delta = 0$, the dynamical system is memoryless; if $\Delta = 1$ (in discrete time) the system is Markovian. Therefore, for a system with a finite memory span, the past is independent of the future. Σ is Δ complete $(\Delta \in T, \Delta > 0)$ if $\{w \in B\} \Leftrightarrow \{(\sigma^t w)|_{[0,\Delta]} \in B|_{[0,*]} \forall t \in T\}$.

Dynamical systems with latent variables and Input/Output Dynamical Systems can be defined in an analogous fashion as before. One method of representing latent variables is through state variables. A state-space dynamical system is defined as a dynamical system with latent variables, $\Sigma = (T, W, X, B_s)$ with $X \subseteq L$, such that the full behavior $B_s \in W \times X$ satisfies the axiom of state. In this case the latent variables, the states, contain sufficient information about the past so as to determine future autonomous behavior. The behavioral equations, such as difference or differential equations, lead to **representations** of dynamical systems.

3.2 *Representations*

The model is defined by the behavior that it allows. The behavior can be defined by a set of inequalities or equations. The structure of the equations is a representation of the model.

A representation that is only a function of current and past signals (outputs) and is called an autoregressive (AR) representation and can be written as $R(\sigma, \sigma^{-1})w = 0$. Where

$$R(s, s^{-1}) = R_L s^L + R_{L-1} s^{L-1} + \ldots + R_{l+1} s^{l+1} + R_l s^l$$

If the system that we are trying to model suggests latent variables to describe the behavior, the autoregressive representation can be expanded to include a moving average part of the past latent variables resulting in an autoregressive-moving-average (ARMA) representation. In this case, the behavioral difference equations relate the time-series of the manifest variables $w : Z \rightarrow R^q$ to the time-series of the latent variables $l : Z \rightarrow R^q$. With appropriate definitions, the ARMA system is defined as:

$$R(\sigma, \sigma^{-1})w = M(\sigma, \sigma^{-1})l$$

[3] Here $\wedge$ denotes concatenation

An important class of ARMA systems are those where $R(s, s^{-1}) = I$. This yields a moving average (MA) representation: $w = M(\sigma, \sigma^{-1})l$

We can combine the above constructs to define a class of models with all of the advantages of completeness – described by the difference equation; state form – the memory is displayed through the latent variables; and nonanticipating Input-Output – an explicit cause and effect structure. This representation is an Input/State/Output representation and is the model class most amenable to analysis, synthesis and simulation.

3.3 *Controllability and Observability*

In a controllable system, the past trajectory does not have a lasting influence on the far future. Sooner or later, any other trajectory, within the controllable subspace, can be attained. In an autonomous system, the past trajectory determines its future completely. Consequently, the lack of controllability implies predictability. As we develop the capability to better understand and control our environments, our ability to predict that environment can suffer. We are limited in our ability to predict by our ability to observe. All dynamical systems are not controllable.

While controllability is intrinsic to the dynamic system, observability is also a function of the representation of that system. This comes about because observability is only an issue for dynamical system model **representations** that have latent variables (by definition, if the variable is a manifest variable it is observed) and is a property where an unobserved signal can be deduced from one which is observed.

3.4 *Discrete-Event Systems (DES)*

The above framework is consistent with the formalized discrete-event systems in theoretical computer science. The behavior is similar to the formal language, a state-space system is like an automation, latent variables are replaced by production rules, interconnections are communications. The most significant difference is the lack of behavioral models (equations) in the theory of DES. Also completeness is usually violated in a DES by initiation and termination rules for event strings.

Since the DES is not complete, representation of these systems requires special consideration. We will see that completeness is required to represent a dynamical system by a behavioral difference equation. Results for representation of complete systems may be generalized to a class of noncomplete systems (including DES) that meet specific restrictions.

A linear time-invariant dynamical system (Z, R^q, B) is called an l_2-system if B is a linear shift-invariant closed subspace of $l_2(Z; R^q)$. Define $\overline{B^{pc}}$ as

the closure of B with respect to the topology of pointwise convergence. With these definitions, results for complete systems may be generalized to l_2-systems satisfying $B = \overline{B^{pc}} \bigcap l_2(Z; R^q)$[4].

4 Metamodeling Issues

With a framework established to characterize system models, we now address the key issue of the inverse modeling problem: "What properties of the behavior allow the system to be represented by a difference (or differential) equation of a particular type?" Analysis of these properties will result in rules and constraints for the setup and design of metamodels. Since we are no longer fitting data but identifying systems, the data used to identify the system must meet certain prescriptions. Explanation and proof of the following statements can be found in the references.

1. To represent a system by means of a difference equation it has to be complete (it cannot have initialization or termination conditions at $t = \pm\infty$) with a finite memory span so that observation of a trajectory on a finite time interval allows conclusions about past behavior independent of what will happen in the future.

2. For a system to be described by AR-equations it must be linear, complete, and time invariant.

3. Since a dynamical system containing latent variables can be converted into an AR representation with an increase in the lag, representation of a dynamical system with latent variables must also be linear, complete, and time invariant.

4. If the dynamical system is controllable (if it is possible to eventually steer the system to a desired trajectory) then the system will also allow an MA representation.

5. An Input-Output dynamical representation can be defined if, and only if, it can be described by an AR-system of behavioral equations $P(\sigma, \sigma^{-1})y = Q(\sigma, \sigma^{-1})u$ with $P(s, s^{-1}) \in R^{p\times q}[s, s^{-1}]$, $Q(s, s^{-1}) \in R^{p\times m}[s, s^{-1}]$ and $\det P \neq 0$.

6. The Input-Output dynamical representation will be nonanticipating if, and only if, $P^{-1}(s, s^{-1})Q(s, s^{-1}) \in R^{p\times m}(s)$ is a matrix of proper rational functions.

7. The manifest behavior of the state variable (an ARMA) representation will belong to L^q. Consequently, every system $\Sigma \in L^q$ admits a finite-dimensional state representation, allows a componentwise I/O representation, and consequently admits an Input/State/Output representation.

[4] See [7] and [8] for definitions of l_2 and L^q spaces.

4.1 *Identifiability*

Identifiability relates to the ability to reconstruct the dynamical laws of the system from a given set of measurements [9]. There are several obstructions to identifiability. Feedback makes it difficult to separate system dynamics from the dynamics of feedback. Structured inputs can interfere with the structure of the behavior. Lastly, the failure of the input to excite all of the modes will prevent observation (and subsequent identification) of the unexcited modes.

Any unstructured input will be sufficiently rich to observe a controllable system. Structured inputs will allow observation and identification if the AR relations defining the structure of the input have large lags that do not interfere with the structure of the system. In other words, if the structure of the input is not seen by the system.

In order to identify a portion of a system, we must be able to observe the response. Observability specifies the ability to determine the trajectory of latent variables from the manifest set. Since controllability allows an MA representation, and any controllable MA representation can be converted into an AR representation by increasing the lag, complete controllability implies observability. Lack of controllability, however, does not imply lack of observability [10]. For systems that can be reduced to an AR-representation, $R_1(\sigma, \sigma^{-1})w_1 + R_2(\sigma, \sigma^{-1})w_2 = 0$ with $R_1[s, s^{-1}] \in R^{g\times q_1}[s, s^{-1}]$ and $R_2[s, s^{-1}] \in R^{g\times q_2}[s, s^{-1}]$ then w_2 is observable from w_1 if, and only if, the rank of the matrix $R_2(\sigma, \sigma^{-1})$ is equal to $q_2 \forall \sigma \neq 0$.

This is why inverse modeling or System Identification is so difficult – the system and our selection of a representation is critical in that it constrains the behaviors of the model, affects our ability to observe latent variables, and impacts our ability to represent the outcomes U.

4.2 *Representing Discrete Event Systems*

The discussion above introduced the issues associated with Discrete Event Systems. Most of System Identification is formulated on continuous, discrete, or continuous-discrete dynamical systems. Many of the simulations are discrete event or connected discrete-event dynamical systems. The question arises: "When can a DES be described by a

difference equation?"

Since completeness is usually violated, this impact must be expressly considered. If a linear time-invariant system is not complete, then whether or not $w : Z \rightarrow R^q$ belongs to the behavior depends on $w(t)$ at $\pm\infty$. However, results for complete systems can be generalized if the system behavior is restricted to a finite dimensional sequence. From Section 3.4, every behavior $B \in L^q$ allows an AR representation. Define a DES as a time-invariant system $\Sigma = (Z, W, B)$ with W a finite set. A DES is internally finite if it can be realized by a finite automation, if there exists a state-space representation of it with a finite-state space. An internally finite and complete DES $\Sigma = (Z, W, B)$ can be described by a behavioral difference equation $f \circ (\sigma^L w, \sigma^{L-1} w, \ldots, \sigma^l w, w) = 0$ for some $L \in Z$ and some $f \circ W^{L+1} \rightarrow \{0, 1\}$.

4.3 *Existence of a true Input-Output Relationship*

Assume that we have observed the input and output of a system and computed a set of linear differential and/or algebraic equations from this data. Have we identified the system? Do these equations establish a true Input-Output relationship suggested by this identification? Answers to these questions are provided by two sequences of subspaces, one in the input space u and the other in the output space y [11].

Consider a system of linear ordinary differential and algebraic equations with constant coefficients: $A(\sigma)\xi + B(\sigma)u + C(\sigma)y = 0$ where (σ) denotes differentiation (or the shift operator for discrete time systems), and ξ contains all of the latent variables not present in the input and output spaces. $A(s)$, $B(s)$, and $C(s)$ are polynomial matrices.

We say that y processes u if the linear space of trajectories $\{y|(y, 0) \in B\}$ is finite dimensional. Therefore, y processes u if u determines y up to a finite number of constants. Also, u is free if for every trajectory u there exists a trajectory y such that $(y, u) \in B$.

Recall that if the dynamical system with latent variables $\Sigma_f = (Z, R^q, R^d, B_f)$ is linear time invariant and complete, the manifest system which it represents $\Sigma = (Z, R^q, B)$ is also linear time invariant and complete. Consequently, for a linear time invariant and complete system, any behavior given by $A(\sigma)\xi + B(\sigma)u + C(\sigma)y = 0$ can also be represented by:

$$B = \left\{ \left| \begin{bmatrix} y \\ u \end{bmatrix} \right| [R_1(\sigma) \;\; R_2(\sigma)] \begin{bmatrix} y \\ u \end{bmatrix} = 0 \right\} \qquad (2)$$

The behavior of such a set of equations stems from an Input-Output system if both conditions of the following proposition hold.

Proposition 1 *Let a behavior B be given by equation 2. where $[R_1(\sigma) \;\; R_2(\sigma)]$ is a polynomial matrix of full row rank. The following statements hold:*

1. y processes u if, and only if, $R_1(s)$ has full column rank
2. u is free if, and only if, $R_1(s)$ has full row rank.

Therefore, $R_1(s)$ must be invertible and the transfer matrix of the system is defined by $T(s) = -R_1^{-1}(s)R_2(s)$

Also, once the identification is accomplished, the subspaces generated by the system (equation 2) can be checked to determine if a true Input-Output relationship has been found (see [11]).

4.4 *Metamodeling Combat Simulations*

With respect to Metamodeling Combat Simulations, the systems we are trying to identify are complex, nonlinear, time varying discrete event systems. In general, for this case, the predictor function is a nonlinear function of past observations and there are too many possibilities for unstructured "black box" models. **Knowledge of the nonlinearities must be built into the model** [12].

Fortunately, in this case, we have explicit knowledge of the nature and characteristics of the model. We have the model (the simulation) that applied the system to the inputs to generate the outputs we are interested in. Given this information, we can build the nonlinearities into the structure of the metamodel and provide the capability to generate a reduced order approximation of the original model. This fact makes Metamodeling as a method of model abstraction feasible. We will exploit this fact to the fullest extent possible.

Care must be taken in the setup of the Metamodeling problem. **The experimental design must provide Input-Output sequences that correctly represent the system structure.** When the metamodel is determined, it is not possible to ask "What is the probability that a particular set of fitted parameters is correct" because there is no statistical universe of models from which the correct one is chosen. There is just one model and a statistical universe of data sets that are drawn from it. It is possible to ask, however, "Given a particular set of parameters, what is the probability that this data set could have occurred?" We can identify the probability of the data given the parameters as the likelihood of the parameters given the data [13].

In addition to the problem setup and experimental design, the metamodel solution comes with limits of its own. Using the space spanned by the original model as the full order model, the metamodel is a reduced order approximation. This reduction inherently limits the span of the manifest (exogenous)

variables associated with the behavior (input or output - if such a map exists). **Consequently, the behaviors allowed by the metamodel will exist within a subspace of the original model.**

Assuming that an Input-Output map exists for the model, **input values will be restricted to a domain** within which the metamodel will be applicable. Outside of this hypersurface, application of the metamodel may provide numbers but will not generate an output that is representative of the modeled system. Also, assuming appropriate inputs, the **output of the metamodel can only be guaranteed to be approximately correct.** As a projection, the metamodel will not contain all of the detail of the original model. There are output error bounds that are a function of both the Metamodel and the input.

Military Engagement Simulations usually are defined to represent real-world events that have a beginning and an end. **Given that the simulation terminates naturally, results for complete systems can be applied** since the system behavior is restricted to a finite dimensional sequence.

In general, the **axiom of state applies** because the simulation is set up in such a way that the initial conditions contain sufficient information about the past so as to determine future autonomous behavior.

Also, the presence of input and output files indicates that an **Input-Output structure with causality** is assumed in the simulation.

5 APPLICATION

5.1 *Introduction to TERSM*

Using the above framework, we applied System Identification to the TERSM Metamodeling problem. TERSM (Tactical Electronic Reconnaissance Simulation Model) was designed to provide comparative performance evaluations of single and multiple-pass DF (direction-finding) systems. The outputs of the simulation are the number of bearing measurements made on each emitter, and the lower bound of the Circular Error Probable (CEP) of the emitter locations. Inputs to the simulation include sensor parameters, an emitter environment, and aircraft parameters. The program functions as shown in Figure 1.

From this overview, you can see that the simulation is centered on the DF sensor and the operations that have to take place within the system. As a result, data generated by the simulation is event driven as a function of DF sensor processing. The update of the simulation time and corresponding motion of the aircraft platform is a function of the time required to process the data. This in turn, is a function of the number of channels operating, the number of emitters detected in the channel, the frequency-scanning technique (continuous serial scanning, parallel scanning, and either serial or parallel priority scanning), and the channel capacities.

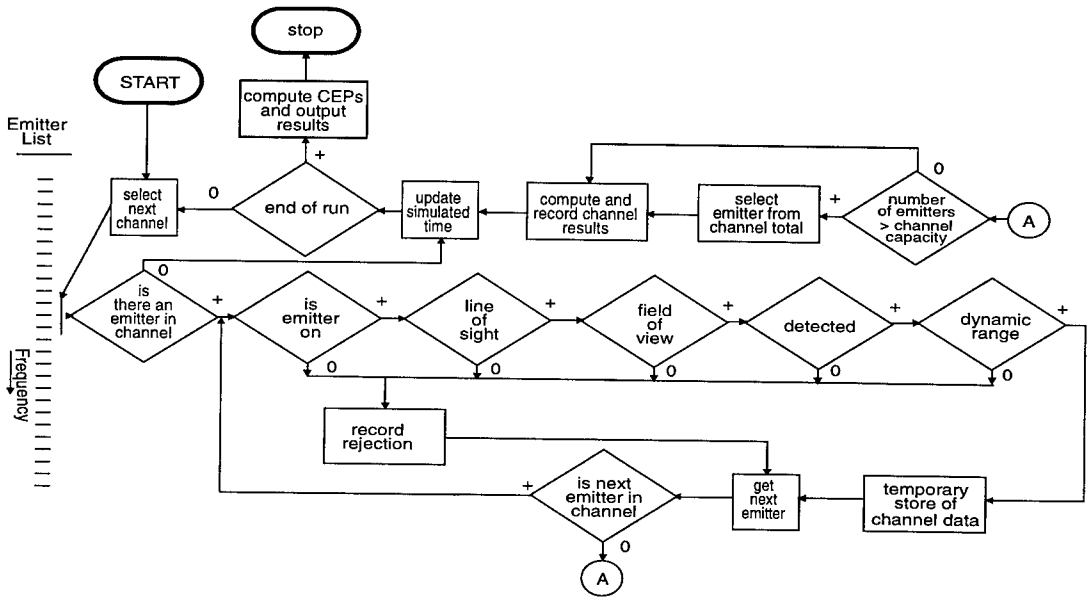


Figure 1: TERSM Functional Flow

The simulation produces a lower bound on emitter location accuracy based on the assumption that the bearing errors from measurement to measurement on the same emitter are normally distributed. During the simulation, the information matrix of the probability density function is computed from the following sums:

$$\begin{aligned} I_{11} &= \sum (y_i - y_e)^2 / r_i^4 \sigma \\ I_{12} &= \sum (x_i - x_e)(y_i - y_e) / r_i^4 \sigma \\ I_{22} &= \sum (x_i - x_e)^2 / r_i^4 \sigma \end{aligned} \quad (3)$$

where σ is the standard deviation of the bearing measurement, x_i and y_i are the aircraft coordinates, x_e and y_e are the emitter coordinates, and r_i is the range at each DF measurement. At termination of the run, the covariance of the emitter location is obtained by inverting the information matrix. The covariance matrix is then diagonalized to obtain variances along the major (σ_L) and minor (σ_s) axes of the location uncertainty ellipse. The Cramer-Rao lower bound for the CEP is then computed from:

$$\begin{aligned} CEP &= .674 + .8 \left(\frac{\sigma_s}{\sigma_L}\right)^2 \quad for \frac{\sigma_s}{\sigma_L} \leq 0.5 \\ &\quad or \\ CEP &= .587 (\sigma_s + \sigma_L) \quad for \frac{\sigma_s}{\sigma_L} > 0.5 \end{aligned} \quad (4)$$

5.2 *Emitter Field*

The emitter field for this experiment consisted of 2359 emitters as shown in Figure 2. The aircraft flight path (solid line) is also shown.

. .

Figure 2: TERSM Emitter Field

5.3 *Previous Work*

Previous work with this simulation resulted in a metamodel that provided an estimate of the number of emitters found with a CEP of 5 nautical miles (nm) or less [1]. This metamodel was generated by a Least Squares fit of selected input data and the number of emitters reported with a CEP of 5 nm or less. The inputs were aircraft altitude and velocity, azimuth coverage, and channel capacity of the sensor. These inputs were combined to generate a nonlinear system with 22 inputs (up to the fourth order of a single input) to produce the square root of the number of emitters with a CEP of 5nm or less. Using a 2 layer Central Composite Experimental Design, the following model was obtained:

$$\begin{aligned} \sqrt{y} &= 23.567 - 0.669x_1 - 2.842x_2 + 1.298x_3 + \\ &\quad 3.344x_4 - 0.491x_1x_3 + 0.963x_1x_4 + \\ &\quad 0.414x_2x_3 + 1.155x_2x_4 + 0.231x_3x_4 + \\ &\quad 0.404x_1x_2x_3 + 0.198x_1x_2x_4 - \\ &\quad 0.285x_2x_3x_4 + 2.037x_1^2 - 0.788x_3^2 + \\ &\quad 0.201x_1x_3x_4 - 2.743x_4^2 + 0.714x_1^3 + \\ &\quad 5.836x_2^3 + 0.744x_3^3 - 2.947x_1^4 - 5.823x_2^4 \end{aligned} \quad (5)$$

where

$$\begin{aligned} x_1 &= Altitude \\ x_2 &= Velocity \\ x_3 &= Azimuth \\ x_4 &= Channel\ Capacity \end{aligned}$$

The model provided an excellent fit with an R^2 of 98.9%, Maximum Absolute Error of 73.51 emitters, and an Average Absolute Relative Error of 4.7%. This is a good example of a Metamodel that can be used to explore the effect of the different input variables on the output via Surface Response Methodology or Capability Based Analysis. It has not, however, identified the system simulated by TERSM. As such, it's utility for simplifying or coupling simulations, further analysis, or updating knowledge bases in expert systems is limited. Also, the domain of validity (range of the response) is not guaranteed outside of the area of the fit.

5.4 *A New Metamodel for TERSM*

We will pursue a different approach in line with the framework suggested in Section 3. Rather than combine inputs to fit the output, we will use our knowledge of the system to identify what is essentially a reduced order model. This model will concentrate on identification of the latent variables inherent to the system. Consequently, the validity of the model will not be restricted to the data used to build it.

We assume that the desired result remains the number of emitters with a CEP of less that 5 nm. Therefore, we will build a Metamodel that will actually compute the CEP (as opposed to other measures that may be of interest such as the probability of detection, the probability of location within 5nm, etc.).

To use this Metamodel for the kind of analysis discussed in [1], however, we would have to relate the input variables of altitude, airspeed, azimuth, and channel capacity to the latent variables used for this identification. This would require running the simulation to capture the data or another mapping from the input variables to the latent variable inputs to this Metamodel.

5.5 *Simulation Parameters and Output*

TERSM was initialized to simulate a single aircraft flying for 1476 seconds at 40,000 feet, 560 knots, with a sensor that could view on both the left and right. The sensor used a parallel scan over 5 bands, scanning from $60MHz$ to $18GHz$, with a channel capacity of 20 emitters, a 90 degree viewing angle, and a 40 degree depression. Data from this simulated flight was collected by slightly modifying (to add a few parameters) existing write statements that already existed in the simulation.

In the simulation, 949 of the 2359 emitters were detected by the sensor. There were 12981 DF measurements. Of these, the sequence of calculations in 2 cases caused the lower bound for the CEP to exceed the capability of the computer and were thus undefined. Three of the individual measurements resulted in estimates of the CEPs in excess of 5000nm. These outliers were removed from the data resulting in 12976 data points. (Note that in TERSM, these calculations did not pose a problem. Since the information matrix was not inverted until the end of the run, the numerical effect of these single measurements were not observable.) For the 12976 data points, the average lower bound for the CEP for the first half of the data points was 140nm, the average lower bound for the second half of the data was 32nm resulting in an overall average of 86nm. The maximum CEP in the data was 4998nm and the minimum was 1.2nm. Of the 949 emitters that were detected, 329 were located with a lower bound on the CEP of less than 5nm.

5.6 *Metamodel Structures*

At this point, two Metamodels were considered. First, a dynamical model of the simulation could have been developed that incorporated both current and past inputs and outputs in a state-space or Box-Jenkins model structure:

$$y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t) \tag{6}$$

Based on the physics of the situation, this would clearly be the most accurate model. In fact, given that the information matrix is a simple sum of inputs, a Markov model should be possible. To generate this Metamodel, the measurement data would be collated for each of the 949 sensors (with up to 58 measurements each). Data for each sensor would be used to recursively identify an Autoregressive model that included past as well as current values of inputs and outputs. This Metamodel could be used in the situation when the state of the sensor (number of measurements for a given emitter, current and past measurements, and estimates of the CEP) is known.

The second Metamodel considered did not attempt to actually model the DF estimation process. This Metamodel was a little more abstract and used the running total of the elements of the information matrix as inputs. In this way, the state of the sensor is fully incorporated in the input data and an Output Error model structure could be used:

$$y(t) = \frac{B(q)}{F(q)}u(t) + e(t) \tag{7}$$

The Output Error structure was selected. First, it used data that was directly available from TERSM without additional processing. Second, it would be simpler to use as a module in another (larger) simulation because the state of the sensor is not required and it more closely resembles the level of abstraction in previous work. Third, the resulting model would be of significantly lower order and consequently contain fewer degrees of freedom to fit the data. Good results would be more difficult to obtain. At first this may not seem logical. But, the purpose of the research was to understand the process of Metamodeling, not to Metamodel a particular simulation. Errors in the process of identification are much more evident in lower order models.

5.7 *Metamodels and Results*

From the discussion on TERSM and analysis of the code, we see that the CEPs are functions of the relative difference between the aircraft and emitter position and the standard deviation of the bearing noise. These parameters are used to calculate the four terms that make up the information matrix. In TERSM, a running total of the information matrix is maintained for each emitter and the lower bound of the CEP is calculated at the end of the run.

The first Metamodel was based on the three primary inputs: the number of DF cuts available to the sensor, the range along-track, and the cross-track range. CEPs calculated with this model were not

very accurate. The Maximum Absolute Error was 4984nm, the Average Error was 166nm, and Average Absolute Relative Error was 385%. Using this data, 719 emitters with a lower bound on the CEP of less than 5nm were found. Adding range as an input reduced the number of emitters found to 670. In all cases, there was considerable correlation in the residual terms.

At this time, 1/range, and the combinations of ranges that were actually used to compute the information matrix, were added to the input bringing the total number of inputs to 6. While the errors for the additional inputs improved, there was still considerable structure in the residuals. Final performance, the number of emitters with a lower bound of the CEP, only improved slightly to 653 (well above the actual number of 329).

Given a known system, every projection of that system into a subspace will reduce the information content of the observed behavior. The only exception is the situation where the kernel of the projection coincides with the null space of the behavior. In the usual case of inverse modeling, the structure of the system is not known. However, if the dynamics of the system are available (as in a simulation) or can be assumed, the number of processes present in the interconnected system can be determined.

During the Metamodeling (inverse modeling) process, it is imperative to model only one system. Otherwise, behaviors associated with both processes will be aliased – preventing the identification of either.

From the above discussion, it is obvious that the TERSM output is the result of two separate processes. This is an important issue. In reality, the CEP should be a piecewise continuous function of the number of measurements and the angular separation of the measurements – a single process based on the geometry of the aircraft and emitter and the statistics of individual measurements. The simulated model in TERSM, however, was a discontinuous function of the uncertainty ellipse. Since the purpose of the metamodel is to represent the simulation, two separate systems had to be modeled.

Based on the value of $\frac{\sigma_s}{\sigma_l}$ (equation 4), the input data was separated. Two identifications, one for each system, were accomplished. The results were immediate, the cross correlation between the output and inputs were within limits, the Maximum Absolute Error was 387nm, the Average Error was .3656nm, the Average Absolute Error was .8508nm, and the Average Absolute Relative Error was 2.9%. Using this data, there were 301 emitters with a CEP lower bound less than 5nm. This Metamodel was based on the (first) half of the data that had an average CEP of 140nm. The remaining data was used to determine the quality of the model.

Since the interest was in emitters with a CEP of 5nm or less, another identification was accomplished using data from the second half of the simulated run. During this portion of the profile, the sensor has more data and can provide a better estimate of the emitter location. The average lower bound for this half of the data was 32nm. Again, the residuals for each system were within limits. For this Metamodel, the Maximum Absolute Error was significantly less at 142nm. This improvement came at the cost of a slight bias with a higher Average Error of 1.1nm and Average Absolute Error of 1.2nm. However, the Average Absolute Relative Error was .8%; also significantly less than the Metamodel based on the first half of the data.

The range of the data makes visual presentation of the results difficult. If all data points are plotted, it is not possible to determine a difference in the actual and Metamodeled data. Figure 3 is a plot of the last 976 data points that had a lower bound of the CEP between 5 and 6 nm – a range that is much less than the average of the data (32nm during this portion of the simulation). Aggregate results with this Metamodel were even better, of the 329 emitters with a CEP less than 5nm, this model predicted 326.

6 Results and Conclusions

This paper provided a framework for the application of System Identification techniques to develop Metamodels for tactical simulations used by the Department of Defense. Use of this framework was successfully demonstrated by the development of a Metamodel for the "Tactical Electronic Reconnaissance Simulation Model."

Issues identified under Metamodeling Combat Simulations must be explicitly addressed: nonlinearities must be built in; the experimental design must isolate the system for identification (identification of multiple independent systems results in an ill-posed problem that fails mathematically); the domain of the input must be relevant to the issues that are to be addressed by the Metamodel; and finally, the accuracy of the model must be adequate to meet *a priori* requirements.

Metamodeling simulations, as opposed to data from an unknown source, provide the ability to structure the experimental design so that very accurate identification of the system or systems is possible. This knowledge also allows control over the input to the Identification algorithm so that the domain and range of the Metamodel can be controlled.

In addition to meaningful results for each Metamodel, the Identification of system models brings with it a large library of research generated to advance the Estimation and Control of linear and nonlinear systems.

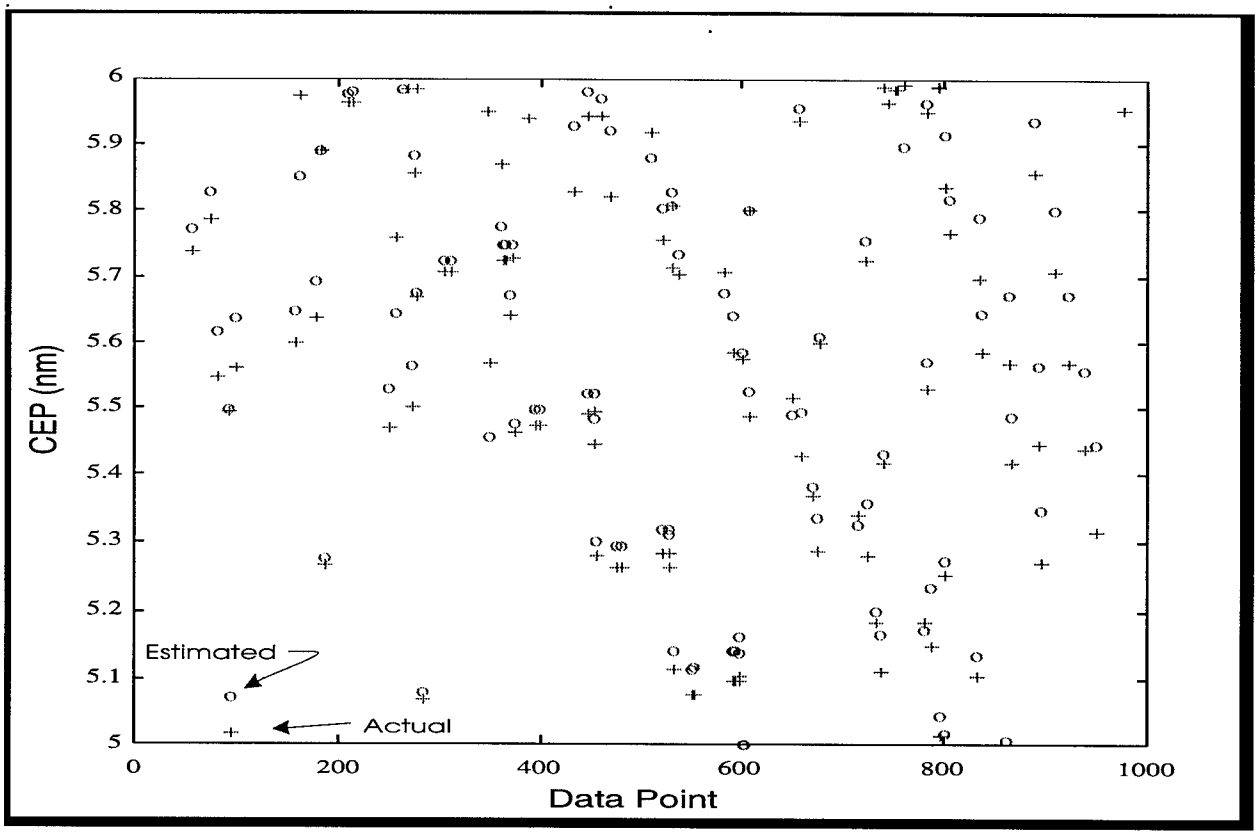


Figure 3: Actual and Metamodel Data for the last 976 data points that were between 5 and 6 nm CEP

REFERENCES

[1] M. A. Zeimer, et. al., "Metamodel Procedures for Air Engagement Simulation Models," *IRAE Technical Report*, Jan 1993.

[2] A.F. Sisti, "Large-Scale Battlefield Simulation Using a Multi-Level Model Integration Methodology," *RL-TR-92-69*, April 1992.

[3] V. Vemuri, *Modeling of Complex Systems*, Academic Press 1978.

[4] J. C. Willems, "Paradigms and Puzzles in the Theory of Dynamical Systems," *IEEE Trans. on Automat. Contr.*, vol. 36, no. 3, pp. 259-294, March 1991.

[5] B.P. Zeigler, "Hierarchical Modular Modeling/Knowledge Representation," *Proc. 1986 Winter Simulation Conf.*, pp. 120-137, 1986.

[6] A.F. Sisti, "A Model Integration Approach to Electronic Combat Effectiveness Evaluation," *RL-TR-89-183*, October 1989.

[7] .A. W. Naylor, and G. R. Sell, *Linear Operator Theory in Engineering and Science*, New York: Springer Verlag 1982.

[8] H.L. Royden, *Real Analysis*, Macmillan Publishing Company, 1988.

[9] T. Kailath, *Linear Systems*, Prentice-Hall, 1980.

[10] H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*, Wiley-Interscience, 1972.

[11] M. Kuijper and J. M. Schumacher, "Input-Output Structure of Linear Differential / Algebraic Systems" *IEEE Trans. on Automat. Contr.*, vol. 38, no. 3, pp. 404-414, March 1993.

[12] L. Ljung, *System Identification: Theory for the User*, Prentice-Hall, New Jersey, 1987.

[13] Press et all, *Numerical Recipes* , Cambridge University Press, 1986.

VERIFICATION, VALIDATION, AND ACCREDITATION (VV&A) OF MODELS AND SIMULATIONS THROUGH REDUCED ORDER METAMODELS

Don Caughlin

Mission Research Corporation,
Colorado Springs, Colorado 80903, U.S.A.

ABSTRACT

This paper provides a new approach to support Verification, Validation, and Accreditation (VV&A) of models and simulations. The need for efficient and objective methods to verify, validate and accredit models and simulations is greater than ever. More and more decisions are based on computer generated data that are derived from models and simulations. The strength of these decisions is a direct function of the validity of this data. Based on the system identification of reduced order models, this new approach approximates a complex high-dimensional model or simulation by a relatively simple mathematical model valid over a specified domain and range of interest. Verification or validation is then accomplished by the straightforward comparison of the reduced order model structure and coefficients with the baseline data or system. Well-developed identification methods and a structured procedure make this process more efficient and objective than existing methods.

1 INTRODUCTION

Increasing computational capability combined with the rapid response and inherent flexibility has allowed M&S to replace some of the more conventional design and analysis methods. Also, our desire to more accurately represent detailed system behavior or to represent "systems of systems" has lead to highly complex models and simulations. These trends, combined with the increased use of M&S by decision makers and designers, demand that M&S results be correct. Yet, as our ability to model the real-world grows, our ability to verify or validate these models shrinks.

As the reliance on M&S continues to grow, the issue of Verification, Validation, and Accreditation (VV&A) takes on increasing importance. With respect to the overall issue of VV&A, there are two competing requirements. First, the decision makers need answers they can trust. This requirement lends itself to strict configuration control where a limited number of accredited models form the body of analytical tools. However, if we restrict our use of models and simulations to those that are accredited, how do we encourage innovation on the part of analysts, accommodate new questions, or respond to the ever-changing environment?

This leads to the second requirement. Decision makers must be able to answer specific questions about very complex environments and phenomenon. This requires a large body of techniques that can be appropriately applied to the specific situation. It also requires an innovative VV&A process that allows independent development while maintaining the validity of the results.

The capability that is lacking is the ability to clearly and efficiently compare a model or simulation with the phenomenon it is supposed to represent or to compare two different interpretations of the real-world. Reduced order metamodels provide this capability and a new approach to support VV&A of models and simulations. Although directed primarily at constructive (man-not-in-the-loop) models, the technique discussed here can also support the Distributed Interactive Simulation (DIS) environment.

The paper is organized as follows: Section 2 provides background on VV&A, definitions for common understanding, and introduces reduced order metamodeling; Section 3 demonstrates how to apply reduced order metamodeling to the VV&A process; Section 4 provides an example of the verification of two versions of the same simulation; and Section 5 summarizes the paper.

2 BACKGROUND

One of the major users of M&S has been the Department of Defense (DoD). DoD has long recognized the importance of M&S and with reduced budgets has

become even more reliant on M&S. This increased reliance, and a concern for the proliferation of models and simulations, has led the Secretary of Defense to direct that each DoD Component shall establish VV&A policies and procedures for M&S applications managed by the DoD Component. Also, the "DoD M&S Executive Agent" shall establish VV&A procedures for their applications.

Current VV&A processes, however, are complex, time-consuming, expensive, and cannot handle the workload generated by the above directives. Consequently, there is insufficient time and money to accredit the models that deserve such status. Furthermore, the process can take so long that changes are often made to the model or simulation before the VV&A process is finished, again drawing the results into question.

The solution to this problem is a consistent, coordinated, requirements-based policy and the ability to efficiently analyze models and simulations. Both of these elements are required. Even with the best policy, it is not possible or desirable to "completely" accredit every model or simulation in existence. This is clearly a poor use of resources. Only models and simulations that need accreditation, for one purpose or another, need to go through this process. Given that we have such a policy, how does one go about the VV&A process so that by the time the simulation is accredited it is still relevant? This paper focuses on a technique to efficiently support verification and validation.

Standard VV&A techniques are not robust and still leave room for interpretation. They generally involve looking at the elements of the model or simulation, dissecting it, and coming to conclusions by analyzing these elements. If we cut a complex problem into smaller more manageable pieces while maintaining the overall complexity, we really do not reduce the overall complexity of the problem that were are trying to solve. We just make it tractable. If we have a complex model, analyzing each and every piece does not make the overall analysis less complex.

This paper provides an alternative solution to this paradigm that will allow the VV&A process to meet the competing requirements and workload demands. This technique is cost effective, timely, and objective. Rather than look at the parts of the model and attempt to integrate the results, we look at the whole model or simulation and identify its ability to represent the behavior of the phenomenon we are interested in.

We do not maintain the overall complexity of the model or simulation. We propose that the analysis of the model or simulation be accomplished via aggregation of the model details into a more manageable piece that has a reduced order (more abstract) representation. This is accomplished by increasing the level of abstraction (reducing the order) of the model or simulation until it is consistent with data used to define the model or simulation. This reduction provides the ability to clearly and efficiently compare a model with the phenomenon it is supposed to represent or to compare two different interpretations of the real-world.

Since reduced order metamodels provide this aggregation and abstraction, we provide a possible solution to the VV&A dilemma. Our technique provides the opportunity to verify or validate a model in a very short period of time, with few resources, and with objective results. With this capability, it is also possible to verify and/or validate (without going through a formal validation process) models or simulations developed to adapt existing models and simulations to new circumstances.

2.1 Definitions

We begin with some definitions to clarify our views on the relationships between models and simulations, verification, validation, and accreditation.

2.1.1 Models and Simulations

A simulation can be defined an instantiation or realization of a model. In this case, the simulation is different from the model. We will use a more abstract definition.

To begin with, a model is a method of expressing a theory. The expression of the model – its representation – distinguishes classes of models. A model can be physical, such as a wind tunnel model of an aircraft. It can be conceptual, like the construct of the Bore atom. Also, the model could be a mathematical relationship or a method (algorithm) of expressing that relationship – a simulation. Therefore, we consider a simulation to be a particular representation of a model and will not distinguish between them.

2.1.2 Verification

Verification is the process of determining that a model implementation accurately represents the developer's conceptual description and specifications.

The verification process confirms that the model functions as it was originally conceived, specified, and designed. Here we compare the output of the model to the conceptual description, specifications, or definitions that were used in its development.

There are two elements to verification. If the model is an original development, it must be verified against its design specifications. If the model is a revision, update, or modification of an existing (verified) model, the performance of the model (and its differences) can be verified with respect to the original specifications or to the original model.

2.1.3 Validation

Validation is the process of determining the degree to which a model is an accurate representation of the real-world from the perspective of the intended uses of the model.

Validation addresses the credibility of the model in its depiction of the modeled world. In this case, the model is not compared to the structure from which it is developed, but to the behavior that it is supposed to represent. An important issue in the validation of a model is its level of fidelity. Our understanding of the phenomenon that the model is supposed to represent must be at the same level of fidelity as the model.

2.1.4 Accreditation

Accreditation is the official certification that a model or simulation is acceptable for a specific purpose.

The accreditation process is the procedure followed by the application sponsor that culminates in the determination that the model is suitable and acceptable for its intended application.

We do not specifically address accreditation, only a method to support accreditation through verification and validation.

2.2 VV&A Methods

While the growing need is real, procedures for VV&A have not kept pace. Current VV&A processes generally involve looking at the elements of the model or simulation via a functional decomposition, and coming to conclusions by analyzing these elements or by a direct comparison with other models. This process is complex, time-consuming, expensive, and still subject to interpretation. General methods of VV&A include:

1. Algorithm checks
2. Peer or independent review
3. Computer aided software engineering tools

Verification is usually accomplished by either logical or code verification methods. Validation can be accomplished either by internal measures (structure of the model) or a comparison of the output of the simulation with other (external) data. We discuss each separately.

2.2.1 Logical Verification Methods

Logical verification requires the identification of a set of assumptions and interactions for which the M&S correctly produces intended results. It determines the appropriateness of the M&S for a particular application and ensures that all assumptions and algorithms are consistent with the conceptual M&S. Methods to accomplish this determination are:

1. Documentation review
2. Design walk-through
3. Comparison of specifications to requirements
4. Comparison of design to specifications

2.2.2 Code Verification Methods

Code verification methods require a rigorous audit of all compilable code to ensure that the representations of verified logic have been properly implemented in the computer code. This audit is usually accomplished by one of the following techniques:

1. Sensitivity analyses and stress tests
2. Code walk-through
3. Algorithm checks
4. Automated test tools
5. Mathematical stability across platforms
6. Units check
7. Statistical test design for stochastic M&S
8. Rule-based systems tools

2.2.3 Validation of the Structure

Validation of the structure analyzes the sensitivity of the output to the input data. It attempts to determine how accurately the model represents the real-world. It ensures that the representation(s) is (are) balanced and consistent.

2.2.4 Output Validation

Validation of the output begins with the feasibility of the results. Are they reasonable relative to the inputs? If the outputs are reasonable, they are compared with historical, test, or laboratory data.

2.3 Metamodels

From the above discussion we see that there is no unifying approach to VV&A. The VV&A process uses essentially the same methods that would be appropriate for design of the model. Without a truly independent and unified approach, VV&A has become manpower intensive and is often subject to interpretation. The reliance on subject matter experts makes

the results of the VV&A a direct relation to the capability of the expert, their familiarity with the specific behavior and representation, and the amount of time that they have to complete the process. In addition, VV&A for DIS requires a separate class of experts in that environment (Lewis 1994).

The problem with VV&A stems from the fact that the underlying phenomenon is high dimensional and complex; representation of these systems is difficult. This is why simulation models are often used. The modeler takes the part of the phenomenon of interest that he understands, and develops an algorithm to represent that part of the behavior. Comparison of this part of the phenomenon to the actual occurrence is not always possible.

This is why we propose that part of the VV&A process consists of an aggregate analysis of the model or simulation using a reduced order (more abstract) representation. Metamodeling has the ability to facilitate this type of abstraction (Zeimer, et al. 1993).

2.3.1 Higher levels of Abstraction – Reduced Order Metamodels

A model is a method of expressing a theory and the expression of the model is its representation. Assume that the representation of a particular model is a simulation. As such, the representation is an algorithm that does not have a closed form representation.

The VV&A methods we discussed above are examples of direct verification or validation of this representation. Another approach to verification or validation of this representation is through a more abstract "black-box" approximation of the causal time dependent behavior represented by this simulation – a metamodel.

Metamodels can be used for hierarchical simulation or for analysis. Used to support hierarchical simulation and model reuse, the metamodel is used in conjunction with (coupled to) other simulations or simulation elements. Analytical metamodels are an independent structure that is used to understand and extract information from the model. This analysis can be focused on the VV&A task.

Sometimes metamodeling is confused with sensitivity analysis. Sensitivity analysis is an analysis of the data given the model. It can be used to reduce the order of the model by considering the sensitivity of the output to certain variables. Our approach is similar but different. In our procedure, we are considering the sensitivity of the model given the data, behavior, or the phenomenon we are trying to model.

2.3.2 General Framework

As an abstraction, a metamodel is a projection of the model onto a subspace defined by new constraints or regions of interest. It is a projection of the behavior from a higher order to a lower order subspace – a reduced order model. One of the most important aspects of this projection is the definition of the basis of that subspace; i.e., the definition of the variables that are to be considered.

There are three ways to define these variables. If we are working with an element of a simulation or if we are comparing a simulation to an exercise or some other real-world data, the variables are defined by the data set. If we are comparing the behavior to the concept used to develop the model, that concept defines the variables. If we are going to compare two versions of the same model, we must first determine the important variables by an analysis of the simulation under consideration.

The construction of a reduced order metamodel (selection of the parameters used for the projection) involves: *a priori* knowledge; the data; a set of metamodel structures; and rules to determine the best model to realize the data. There are two basic techniques available for reduced order modeling: direct and inverse modeling.

2.3.3 Direct Methods

First, a reduced order model could be developed by applying basic principles to generate a more abstract (approximate) version of the original model. This would be an example of direct modeling. Direct modeling is characterized by a specification of the elements of the model. Complicated systems are modeled by "tearing" a system into its components, modeling these components in a process called "zooming," and then interconnecting these components to construct a "physical" realization of the system (Sisti 1992, Willems 1991, Sisti 1989). The level of abstraction is controlled by the detail of the specification. The model reveals the structure of the theory and allows the prediction of the response to exogenous inputs as a function of the state of the system. The solution of this modeling problem requires an understanding of the process being modeled and methods to express this understanding **at the desired level of fidelity**.

Reduced order models developed using this technique have been proposed in the VV&A literature (Phase 3 – Concept Validation in Lewis (1994)). They are "standalone" versions – completely new models. The relationship between the real system, the original model, and reduced order model is contained in the

two mappings from the underlying system to each of the models. Figure 1 depicts this correspondence.

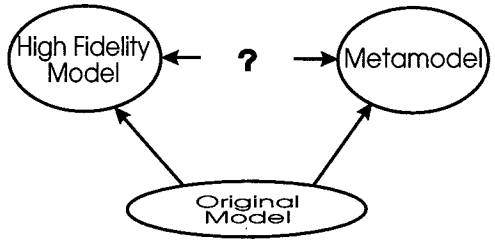


Figure 1: Direct Model Correspondence

As seen from the figure, there is no guarantee that a usable correspondence will exist between the reduced order model and the high fidelity model (Naylor and Sell 1982, Royden 1988). Traceability from the high-fidelity model to the more abstract, lower fidelity, reduced order model becomes a significant issue. Also, this technique still requires an *a priori* understanding of the structure of the elements and the interconnections between these elements at the specific level of fidelity selected. This could be a difficult and risky task and lack of this knowledge is often the reason that a high fidelity simulation was used in the first place.

Since traceability is not guaranteed, this technique does not provide any efficiencies beyond standard VV&A procedures.

2.3.4 Inverse Methods

The second technique develops the reduced order model from the input-output data generated by the original model or simulation. This technique is an example of the "inverse problem," and is represented by Figure 2. From the figure, we see that the correspondence between the original model and the reduced order model is direct. The issues now are the level of fidelity, range of applicability, and accuracy of the response. These are a function of the reduced order modeling technique and data.

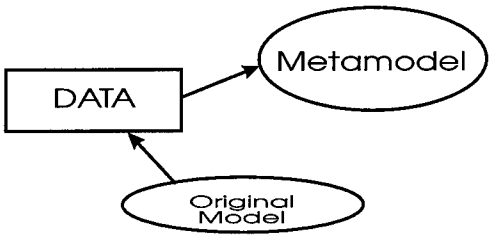


Figure 2: Inverse Model Correspondence

Properly developed, a reduced order model derived from inverse modeling is clearly a mathematical approximation between a set of input factors and responses generated by the high fidelity model. Traceability to the high fidelity model is immediate. As such, it allows the assessment of individual factors on the performance of the model and can be used to study system behavior, verify responses with specifications, or validate the model with respect to real-world data.

3 REDUCED ORDER METAMODELS FOR VV&A

VV&A has many dimensions. Although the procedure is the same, we consider each case separately to facilitate understanding. Assume that we have a reduced order model of an existing simulation and that we also have a similar description of the real-world data that can be used for comparison.

3.1 Verification of an Original Model

In our first case, we have a model that was developed from a specification or conceptual design. Verification is straightforward. We directly compare the reduced order metamodel structure and coefficients to "expected values" inherent in the design specification that came from the real-world experiments, exercises, or test data used to develop the specification.

3.2 Verification of a Modified Simulation

Here we have an existing accredited simulation that has been modified for some purpose (improved execution speed, hosted on a new platform, new capability, etc.). As stated above, we can verify the model with respect to the specifications or, for the portions of the modified simulation that do not add capability, to the existing (unmodified) simulation. If we use the original specification as the baseline, we proceed as above. If we use the existing simulation as the baseline, verification consists of developing a reduced order model of the original and modified simulations using the same model structure. Now, since the structure of each reduced order model is identical, we simply compare the reduced order metamodel coefficients.

3.3 Validation

This is the most complex use of reduced order metamodeling. In order to use reduced order metamodeling to validate a model, we must compare the reduced order model to real-world data. This requires that we have a record of the phenomenon that we have modeled. Also, this record must contain all of

the behavioral characteristics that have been incorporated into the model. Given this record, we develop a reduced order model of both the real-world event and the model we are going to validate. Once we have these reduced models, we simply compare the reduced order model coefficients.

If additional information was included in the model that was based on subject matter expertise or analogy and not available in the real-world data, this additional data must be also added to the real-world data to make the comparison possible.

4 RESULTS AND DISCUSSION

The theory supporting reduced order metamodels has been developed and successful applications have been demonstrated. Zeimer, Tew, Sargent, and Sisti (1993) developed a static least squares metamodel of the Tactical Electronic Reconnaissance Simulation Model (TERSM) that approximated the number of emitters reported with a CEP of 5 nm or less. Caughlin (1994a) outlined a general framework for approaching the reduced order metamodeling problem that would support dynamical system models and presented an output-error dynamical metamodel of TERSM. In Caughlin (1994b) we expanded the dynamics to include Ito stochastic systems and applied an optimization technique (Adaptive Simulated Annealing) to generate a TERSM metamodel that accommodated the stochastic nature of the simulation.

All of the above were examples of analytical metamodels (although the last two could be used as simulation metamodels). The first metamodel addressed the final results of the simulation (in terms of modeled system accuracy). The output-error metamodel approximated the system behavior as represented by the simulation. The third metamodel represented the performance of the system in locating a single emitter and approximated the accuracy of the location estimate as the number of measurements increased.

We now provide a simple example of reduced order metamodeling for verification of a modified simulation (the situation described in Section 3.2 above).

The static least squares TERSM metamodel generated by Zeimer, Tew, Sargent, and Sisti related aircraft altitude, aircraft velocity, sensor azimuth coverage, and sensor channel capacity to the number of emitters located within a 5 nautical mile circular error probable (CEP). This model is shown below:

$$\begin{aligned}\sqrt{y} \;=\; & 23.567 - 0.669x_1 - 2.842x_2 + 1.298x_3 + \\ & 3.344x_4 - 0.491x_1x_3 + 0.963x_1x_4 + \\ & 0.414x_2x_3 + 1.155x_2x_4 + 0.231x_3x_4 + \\ & 0.404x_1x_2x_3 + 0.198x_1x_2x_4 - \\ & 0.285x_2x_3x_4 + 2.037x_1^2 - 0.788x_3^2 + \\ & 0.201x_1x_3x_4 - 2.743x_4^2 + 0.714x_1^3 + \\ & 5.836x_2^3 + 0.744x_3^3 - 2.947x_1^4 - 5.823x_2^4\end{aligned} \tag{1}$$

This model was developed from the Version 1 data (shown in Table 1) that came from simulation runs on a Sun workstation. This simulation was optimized for this workstation and included code to support a RAMTEK display of the emitter field and results.

Another version of the code (Version 2) was recovered from the archive and hosted on a 100 MHz i486 PC using Lahey Fortran 77L EM/32. Answers provided by this version of the simulation were similar but not the same as the results from the experiment run on the Unix workstation. If the original simulation was accredited, could this second representation also be considered a "verified" representation of the tactical electronic reconnaissance system?

Standard VV&A procedures could have been used to answer this question. This would require an extensive analysis of the code, the different compilers, and the effects of the numerical accuracy. Instead, we used reduced order metamodeling. The same conditions that were run on the workstation were duplicated on the PC. The least squares metamodel (using the same model structure) generated from this data is:

$$\begin{aligned}\sqrt{y} \;=\; & 22.4331 - 0.0148x_1 - 2.7822x_2 + 0.1432x_3 + \\ & 3.1432x_4 + 0.3653x_1x_3 + 1.2439x_1x_4 + \\ & 0.1483x_2x_3 + 0.4430x_2x_4 + 0.2698x_3x_4 + \\ & 0.4369x_1x_2x_3 + 0.3286x_1x_2x_4 + \\ & 0.0960x_2x_3x_4 - 0.2791x_1^2 - 0.8326x_3^2 - \\ & 0.7642x_1x_3x_4 - 1.8413x_4^2 + 0.7577x_1^3 + \\ & 4.9038x_2^3 + 1.0924x_3^3 - 1.1907x_1^4 - 4.8443x_2^4\end{aligned} \tag{2}$$

The angular difference between the subspaces defined by the vectors of coefficients is .15 radians indicating that, while similar, the two metamodels contain different information. With the standard assumptions on the data, the probability of error in accepting the hypothesis that both of these models represent the same simulation is approximately 70%. Clearly, the two versions of the simulation do not represent the same behavior.

There are two potential reasons for the differences between the output of the two versions of the "same" simulation. First, it is possible that the experimental procedures were different. Since all of the data sets for the original experiment were not available, one or more of the 53 other parameters used in TERSM to

define the aircraft and sensor performance may have been set in such a manner that the simulated systems were not the same. Correcting the differences in the parameters may result in the same behavior.

If different experimental procedures are ruled out, the simulated systems should be identical. In this case, we conclude that the two simulations are not representations of the same high fidelity model. Version 2 should not be considered a "verified" representation of the Tactical Electronic Reconnaissance Simulation Model.

5 CONCLUSION

In this paper we have presented an alternative approach that will allow the VV&A process to meet the competing requirements and workload demands. This approach does not maintain the overall complexity of the model or simulation, but verifies or validates a simulation through analysis of a reduced order (more abstract) representation of the simulation. By increasing the level of abstraction (reducing the order) of the model or simulation, we aggregate the model details into a more manageable form.

Reduced order metamodeling was then used to examine two versions of the same simulation. The procedure clearly demonstrated the probability of error in accepting the second version of the "same" simulation as representative of the first.

This technique is cost effective, timely, and objective. Increasing the level of abstraction provides the ability to clearly and efficiently compare a model with the phenomenon it represents or to compare two different interpretations of the same behavior.

A reduced order metamodel is a projection onto a lower order subspace. The parameters that define this projection are well defined for simulation and analytical metamodels. Since reduced order metamodeling for VV&A is a new application of this method, further research is required to define the best approach to define the projection parameters.

REFERENCES

Anderson, et al. 1989. SIMTAX, A Taxonomy for Warfare Simulation. Workshop report taken from the *Catalog of Wargaming and Military Simulation Models, 11th Edition*, Force Structure, Resource, and Assignment Directorate (J-8), The Joint Staff, Washington, DC 20318-8000.

Caughlin, D. 1994. A Metamodeling Approach to Model Abstraction. *Proceedings Fourth Annual IEEE Dual-Use Technologies & Applications Conference*, 387-396. SUNY Institute of Technology, Utica/Rome, New York.

Caughlin, D. 1994. An Evaluation of Simulated Annealing for Modeling Air Combat Simulations. *Proceedings Fourth Annual IEEE Dual-Use Technologies & Application Conference*, 412-421. SUNY Institute of Technology, Utica/Rome, New York.

Lewis, R. O. 1994. Verification, Validation, and Accreditation (VV&A) Process for Distributed Interactive Simulations (DIS). *Proceedings of the 10th DIS Workshop on Standards for the Interoperability of Defense Simulations*, 373-379. Institute for Simulation and Training, University of Central Florida, Orlando, Florida

Naylor, A. W., and G. R. Sell. 1982. *Linear Operator Theory in Engineering and Science.* New York: Springer Verlag.

Norton, J. P. 1988. *An Introduction to Identification.* San Diego: Academic Press.

Royden, H. L. 1988. *Real Analysis.* New York:Macmillan Publishing Company.

Sinha, N. K., and B. Kuszta. 1983. *Modeling and Identification of Dynamic Systems.* New York:Van Nostrand Reinhold.

Sisti, A. F. 1992. Large-Scale Battlefield Simulation Using a Multi-Level Model Integration Methodology. *Rome Laboratory Technical Report RL-TR-92-69.*

Sisti, A. F. 1989. A Model Integration Approach to Electronic Combat Effectiveness Evaluation. *Rome Laboratory Technical Report RL-TR-89-183.*

Willems, J. C. 1991. Paradigms and Puzzles in the Theory of Dynamical Systems. *IEEE Transactions on Automatic Control*, 36:259-294.

Zeimer, et al. 1993. Metamodel Procedures for Air Engagement Simulation Models. *Rome Laboratory/IRAE Technical Report.*

AUTHOR BIOGRAPHY

DON CAUGHLIN is Chief Scientist at Mission Research Corporation, Colorado Springs. He received a B.S. in Physics from the Air Force Academy, an MBA from the University of Utah, and M.S. and Ph.D. degrees in Electrical Engineering from the University of Florida. His research interests include system identification, pattern recognition, and intelligent control. Dr. Caughlin has over 27 years experience as an experimental test pilot, research scientist, program manager, and was also Associate Dean of the School of Engineering at the Air Force Institute of Technology. He is a senior member of IEEE and AIAA and a member of the Society of Experimental Test Pilots.

Table 1: Input-Output Data for Metamodel Construction

ALTITUDE	VELOCITY	AZIMUTH COVERAGE	CHANNEL CAPACITY	EMITTERS VERSION 1	EMITTERS VERSION 2
40000	1150	150	30	615	514
40000	1150	150	4	193	158
40000	1150	60	30	327	329
40000	1150	60	4	53	69
40000	186	150	30	247	278
40000	186	150	4	73	73
40000	186	60	30	111	174
40000	186	60	4	47	61
5000	1150	150	30	436	284
5000	1150	150	4	226	183
5000	1150	60	30	322	250
5000	1150	60	4	138	149
5000	186	150	30	180	180
5000	186	150	4	116	94
5000	186	60	30	98	105
5000	186	60	4	66	66
22500	668	105	17	62	519
5000	668	105	17	439	307
40000	668	105	17	570	523
22500	186	105	17	181	210
22500	1150	105	17	464	412
22500	668	60	17	419	414
22500	668	150	17	607	505
22500	668	105	4	240	252
22500	668	105	30	658	617
31250	909	128	24	621	521
31250	909	128	10	424	361
31250	909	82	24	512	489
31250	909	82	10	347	322
31250	427	128	24	634	579
31250	427	128	10	489	399
31250	427	82	24	570	556
31250	427	82	0	434	396
13750	909	128	24	602	486
13750	909	128	10	441	346
13750	909	82	24	560	469
13750	909	82	10	373	339
13750	427	128	24	651	567
13750	427	128	10	526	404
13750	427	82	24	605	535
13750	427	82	10	471	411
13750	668	105	17	580	495
31250	668	105	17	584	504
22500	427	105	17	575	524
22500	909	105	17	529	446
22500	668	82	17	512	499
22500	668	128	17	597	523
22500	668	105	10	441	406
22500	668	105	24	640	585

New Procedures to Metamodel Simulations

Don Caughlin
Mission Research Corporation
Colorado Springs, Colorado, 80903
donc@mrccos.com

Abstract

A metamodel is a mathematical approximation of the system relationships defined by a high fidelity model or simulation. This paper presents new results that expand the set of available metamodel representations beyond the traditional least squares formulation and adds the capability to use dynamical metamodels. These results are supported by a new taxonomy of metamodel structures and methods that allow separation of the metamodeling process into a set of sequential decisions based on a priori information. [1]

1 Introduction

In [1] we introduced a framework for the application of System Identification techniques to develop suitable metamodels for tactical combat simulations used by the Department of Defense. We filled in the framework with concrete definitions and identified specific issues associated with the representation of dynamical systems. Particular attention was given to the discussion of experimental design requirements for metamodeling tactical engagement (usually Discrete Event System - DES) simulations. We demonstrated this approach by outlining the development of an output-error metamodel for the "Tactical Electronic Reconnaissance Simulation Model."

Although this framework was consistent with metamodeling procedures defined in [2], the development of the metamodel required too many decisions to determine the model structure, method of identification, and identification criteria. Each decision was a complex function of *a priori* information and prior selections in the metamodeling process. This paper presents a new approach to support the development of metamodels that is based on a new taxonomy of structures and methods that allows the separation of the metamodeling process into a set of sequential decisions based on *a priori* information.

[1] This work was supported in part by The USAF Rome Laboratory Contract F30602-94-C-0110

The paper is organized as follows: Section 2 introduces metamodels; Section 3 introduces the general approach to metamodeling; Section 4 outlines a new approach to the definition of the problem; Section 5 continues with the general approach and discusses the metamodeling process; Section 6 summarizes the paper with results and conclusions.

2 Metamodels

A model is a structure that can be used for understanding the behavior of a system [3]. Assume that we have a model of a system that cannot be used directly. A solution may not exist, it may be too complicated for a closed-form solution, it may require too much time to numerically determine a particular solution, or it may be a high-fidelity simulation that provides much more detail than we are interested in. Efficient use of this model requires a "black-box" approximation of the causal time dependent behavior of the model – a metamodel.

There are two general metamodeling techniques; the "Direct" and "Inverse" methods. Direct metamodeling is developed by applying basic principles to generate a more abstract (approximate) version of the original model. In this paper we consider inverse modeling and concentrate on the metamodel structures and rules to determine the best model.

2.1 Metamodeling via Solution of an "Inverse Problem"

Inverse modeling begins with the input-output data generated by the high fidelity model or simulation and develops the metamodel (mathematical relationship) from this data. In this case, we have some measure of the input and output response and seek an expression that characterizes the process by which the outputs are generated. This type of problem usually has multiple solutions out of which an acceptable solution must be selected.

In our approach, we are not trying to "fit" data. We are attempting to identify the underlying processes that define the system that generated the data. There-

fore, the focus is not on statistics but on the system theoretic properties of the manifest behavior.

Dynamical systems acquire their importance from the fact that they exhibit memory or the potential to model phenomena where the past influences the future. A dynamical system is a family of trajectories without reference to I/O maps, variables, or behavioral equations. The system is coupled to its environment and is not defined by any associated model.

The metamodel, then, is defined by the behavior it allows. This behavior is represented by inequalities or equations which can be grouped into sets. As we shall see in Section 5, selection of the proper metamodel set is critical to generation of an acceptable solution.

2.2 Metamodeling Simulations

With respect to metamodeling simulations, the systems we are trying to identify are complex, nonlinear, and time-varying. They can be continuous, discrete, or discrete event systems. In general, for these cases, the predictor function is a nonlinear function of past observations, and there are too many possibilities for unstructured "black box" models. Knowledge of the nonlinearities must be built into the model [4].

Fortunately, in our case, we have explicit knowledge of the nature and characteristics of the high fidelity system. We have the model that applied the system to the inputs to generate the outputs we are interested in. Given this information, we can build the nonlinearities into the structure of the metamodel and provide the capability to generate a reduced order approximation of the original model. This fact makes metamodeling as a method of model abstraction feasible. We exploit this fact to the fullest extent possible.

In addition to knowledge of nonlinearities, other requirements must be met to allow representation by a finite dimensional, reduced order approximation: the system must be complete; the axium of state must apply; and the output must be nonanticipating.

Assuming that the underlying system modeled by the simulation is well behaved (Markovian, complete with respect to the modeled behavior), the following is required to metamodel simulations:

1. Data must include the behavior we are to model.
2. The latent variables must be observable.
3. The input must be persistently exciting.
4. For a stochastic system, the ensemble of trajectories must span the space.
5. Any single trajectory must span both the input and output space and be sufficiently long so that the state transition probabilities also span the allowable probability space, and the distribution of these probabilities are the same as the underlying system.

3 General Approach

Reference [2] presented the following metamodeling procedure:

1. Determine the purpose of the metamodel.
2. Identify the response
3. Identify important response characteristics.
4. Identify input factors.
5. Identify important input characteristics.
6. Specify the experimental region.
7. Select validity measures.
8. Specify required validity.
9. Postulate a metamodel based on:
   Input - Output response characteristics.
   Experimental region dimensions.
   Required validity.
10. Select an experimental design.
11. Obtain data.
12. Fit the metamodel.
13. Assess the validity of the model.

The first eight steps of the metamodeling procedure provide the prior knowledge or metamodel requirements that define the problem. The remaining steps define the experimental setup, the model structure, the method of identification, and validity measures used to develop and verify the metamodel.

To streamline the development of techniques for metamodeling simulations, we separated the procedure into two general areas. The first eight became the foundation for the **problem definition**; the remaining steps were grouped in an iterative scheme as the **metamodeling process**.

Therefore, in order to categorize metamodeling problems and their solutions, each of these areas were analyzed to derive a taxonomy that would support the metamodeling procedure.

4 Problem Definition

We define a metamodeling problem as the direct sum of the metamodel requirements and the model (simulation). This means that the same simulation could be part of two different metamodeling problems if the requirements were different. Or conversely, the same set of requirements applied to two different (nonsimilar) simulations also leads to two different metamodeling problems.

Consequently, to define the problem, we must consider both elements of the direct sum – the purpose of the metamodel and the simulation characteristics.

4.1 Metamodel Purpose

As mathematical relationships, metamodels can be developed to support two general purposes: (1) Analysis; or (2) Hierarchical simulation.

First, a metamodel can be used for analysis. In this case, the metamodel becomes an independent structure that is used to understand and extract information from the model.

Secondly, a metamodel can be used to support hierarchical simulation and model reuse. In this case, the metamodel is used in conjunction with (coupled to) other simulations or simulation elements to answer larger questions that are not supported within the structure of the modeled simulation.

This selection defines the metamodel purpose and provides clear boundary conditions for follow-on selections in Steps 2 through 8.

4.2 Simulation Characteristics

Since all of the remaining problem definition decisions are a function (direct sum) of both the metamodel requirements and the simulation that is to be modeled, we concentrate on the aggregate space of simulation characteristics. Research has suggested that both a general (external) description of the simulation or model as well as further detail on the (internal) process structure of the internal components is required [1, 5].

The classification defined by the "SIMTAX, A Taxonomy for Warfare Simulation" was completely adequate for the external description [6]. It is a descriptive framework designed to guide the development, acquisition, and use of warfare models and provides the basis for classifying objects for identification, retrieval, and research purposes.

Selection of a metamodel structure, however, requires detailed information not contained in the simulation and model catalogues. To provide a link between the more general taxonomy outlined above and specific metamodeling techniques, a more detailed internal taxonomy was appended to the SIMTAX. The purpose of this additional detail is to describe the structure of the simulation in terms of system theoretic definitions common to control engineering.

Figure 1 depicts the model of a continuous system with a sampled measurement. In development of a metamodel, we try to isolate and identify each of the individual elements in this model. Consequently, we must be able to characterize the type of processing that takes place in each of the blocks.

Formulating the metamodeling problem with this additional detail is important for two reasons. First, each of these blocks may be represented by a separable process, and it is usually not possible to simultaneously identify more than one process. If we try to simultaneously identify two processes and the processes are independent, a rank deficiency in the uncoupled equations causes numerical difficulties. If the processes are dependent, behaviors associated with both processes will be combined preventing the identification of either. If one is successful in simultaneously identifying multiple processes, performance of the resulting metamodel is usually poor.

Table 1: Internal Processing Description.

Basis	Physics based Event based
System Inputs Outputs	Linear Nonlinear Stochastic Deterministic
Result/Trajectory	Functional Statistical base
Level	SISO MISO MIMO
Process description	Complex Simple Coupled
Interval	Continuous time Discrete time Continuous - discrete time Discrete-event

Categories and selections for these categories that were used to provide the additional detail on the internal structure are shown in Table 1.

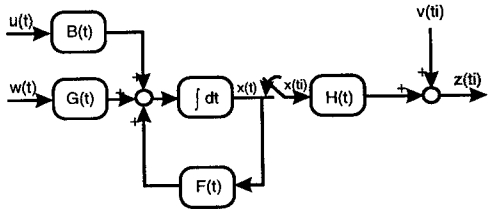


Figure 1: System Model.

5 Metamodeling Process

At this point, we have determined the purpose of the metamodel. In the definition of this purpose, we have identified the input and response that we are interested in and determined the important characteristics of these data. Also for this purpose, we have defined the region of interest, selected validity measures, and specified the required validity.

While there are other issues that must be addressed by the metamodeling process, the remainder of this paper will concentrate on decisions associated with "Step 9: Postulate a metamodel." The completion of this step requires a number of interrelated selections. So many options are available, however, that the combination of model selection, error criterion, identification technique, and numerical methods leads to an overwhelming myriad of "identification methods."

Many specific identification and statistical methods have been developed to accommodate the differences in model structures, data length, measurement error statistics, etc. Also, the literature contains considerable discussion on particular methods with very little discussion on the relationship of these techniques to each other or to a general methodology. The result is a confusing array of unconnected methods with little or no guidance on the application of the techniques to general classes of problems.

Since we are looking for procedures to handle general metamodeling problems, we discuss these methods as elements of a more general structure and have reduced these selections to four that define the model set: **system description**; **system class**; **metamodel structure**; and **identification methodology**.

For any given problem, multiple model sets are available. In each of these model sets, a most powerful unfalsified model will exist (given that the requirements of Section 2.2 are met) [7]. Consequently, the performance of the metamodel will be limited by the match between the metamodel set and actual system that generated the behavior.

5.1 System Description

In the definition of the system description, the first selection concerns the system type that will define the allowed behavior of the models. Here, the most basic questions must be addressed. How are the parameters described? Is the representation going to include dynamics or will it be static? Will the model contain latent variables? If it is dynamic, is it time invariant or time varying?

Is the algebraic structure linear or nonlinear? Are disturbances, noise, and randomness accommodated? Is the system defined as continuous, discrete, continuous-discrete, or as a discrete-event system? Table 2 outlines the possible selections that define the system description.

Table 2: System Description.

Selection	Options
Type	Static Dynamic - Time Invariant Dynamic - Time Varying
Algebraic Structure	Linear Nonlinear
Randomness	Stochastic Deterministic
Time	Continuous time Discrete time Continuous - discrete time Discrete-event

Table 3: System Classes and Representations.

MODEL CLASS	FORMS OF THE REPRESENTATION
SISO MISO MIMO	Polynomial Matrix Fraction State Space

5.2 System Class

In addition to the system description, the class of representation is also needed to define the overall model set. This class is defined by the interaction of the variables and the representation. Table 3 provides a list of the general system classes and the possible form of the representations (see, for example [8, 9]).

Comparing Tables 2 and 3 with Table 1, we see that the characteristics of the behavior we are modeling define the first two elements of the metamodel set: the system description and the system class.

5.3 Metamodel Structure

Once a system description and class that match the underlying behavior have been selected, the next decision is selection of the model structure to use in describing the response of the system to the inputs. There are many metamodel structures, and this area generates much of the complexity in identification.

We simplify this decision by defining two general model structures, **predictor models** and **probabilistic models**. A predictor model only defines the predictor equation(s). Predictor models are models that specify the elements of the transfer function in terms of some parameter set. The models generated

from these structures are deterministic in nature. [2]

A probabilistic model accommodates the fact that many systems are subject to known disturbances that are not (or cannot be) completely categorized. The statistics of the noises and disturbances are to be included as random variables. Probabilistic models supplement the parametric description with a description of the density function (or moments) of the noise (disturbance) that acts on the system. The variables of the system being identified become functions of random variables. In these situations, different realizations of an experiment (simulation run) may not produce exactly the same results. Consequently, the output of a probabilistic model is the conditional expected value and the joint or conditional probability density functions (JPDF or CPDF) of the variables.

The following two subsections discuss these two model structures.

5.3.1 Predictor Models

Static. Static systems can be either linear or nonlinear. The predictor equations for static models are the actual input-output map that comes from the selected representation and are similar to those representing dynamical systems. Also, static models can be set up using dynamical model structures with a zero state transition.

Dynamic. For dynamic systems, system identification requires the ability to use the model structure to predict the output of the model. The differences between this prediction and the actual data are then used to arrive at the parameter set which minimizes the error. As the complexity of the system description increases, the flexibility in the selection of the representation (polynomial, matrix fraction description, state space) decreases.

We will consider three types of dynamic systems: linear time-invariant, linear time-varying, and nonlinear. To save space, discrete realizations are presented. However, continuous realizations can also be used. All nonlinear systems will be assumed to be Markov.

Linear Time-Invariant Predictor Models. There are a number of ways of defining the transfer function (input-output map) associated with linear time-invariant predictor models: polynomial; frequency function; or by it's zeros and poles. These descriptions are most appropriate for SISO systems.

[2]Predictor models, however, do allow for the prediction or measurement error. And since the coefficients were generated via a minimization of some error criterion with assumed statistics, the coefficients will be random variables with an error distribution. Since the estimates are functions of these random variables, this distribution can be used to compute error bounds of the estimate.

MISO systems are best represented by a state space or polynomial format that explicitly defines the coefficients of each of the input and output terms.

Our general linear metamodel structure is:

$$y(t) = G(q)u(t) + H(q)e(t) \tag{1}$$

where $y(t)$ is the output, $u(t)$ is the input, and $e(t)$ is the error. Here q^{-1} is the backward shift operator so that $q^{-1}u(t) = u(t-1)$. Consequently, the polynomials have the form $G(q) = 1 + g_1 q^{-1} + \cdots + g_{n_g} q^{-n_g}$.

From this general model, we can define a SISO or MISO model structure as:

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t) \tag{2}$$

The predictor for this general polynomial structure is:

$$\hat{y}(t|\theta) = \frac{D(q)B(q)}{C(q)F(q)}u(t) + \left[1 - \frac{D(q)A(q)}{C(q)}\right]y(t) \tag{3}$$

Latent variables (that are not past values of the input or output) can also be defined in the polynomial format by augmenting the input-output relationships to include the addtional variables.

MIMO systems are most amenable to the state space format. This format also has the most flexibility in defining the relationship to latent variables. In this description we add the state variable $x(t_i)$ that is propagated forward in time by:

$$\hat{x}(t_{i+1}|\theta) = A(\theta)x(t_i, \theta) + B(\theta)u(t_i) \tag{4}$$

and the measurement equation:

$$\hat{y}(t_i|\theta) = C(\theta)x(t_i, \theta) + D(\theta)u(t_i) \tag{5}$$

that provides the output.

One of the most flexible state space predictor models is the directly parameterized innovations form. Based on the classical steady state Kalman filter, this model accommodates the fact that measurement and process noise are present but does not require knowledge of the disturbance properties. This is accomplished by parameterizing and identifying the Kalman Gain instead of the process and noise descriptions: [3]

$$\hat{x}(t_{i+1}, \theta) = A(\theta)x(t_i, \theta) + B(\theta)u(t_i) + K(\theta)\left[e(t_i)\right] \tag{6}$$

Linear time-varying systems. Linear time-varying systems are restricted to weighting function and state space forms. Predictor metamodels for use with a weighting function have the same form as metamodels used for time-invariant systems except that the weighting function is time varying. Time-varying state

[3]The error $e(t_i|\theta) = y(t_i) - C(\theta)\hat{x}(t_i, \theta)$.

space models are similar to the time-invariant state models with the exception of the time index on the coefficients.

Nonlinear Models. Systems with linear dynamics and static input nonlinearities can be handled by redefining input of the system to exclude this nonlinearity (Hammerstein model). With this new definition, the system can be identified by a linear model.

Nonlinear systems (that are not approximated by linearization or perturbation) are restricted either to a pseudolinear form or state space descriptions. We define the pseudolinear form as $\hat{y}(t|\theta) = \theta^T \phi(t)$ where θ^T is the vector of unknown coefficients and $\phi(t)$ contains the nonlinear combinations (functions) of the input data. Although the structure looks static, dynamics can be included in the pseudolinear model by including nonlinear combinations of past data.

If we want to explicitly consider system dynamics for nonlinear predictor models, there is only one option: a nonlinear state space or simulation model defined as:

$$x(t_{i+1}|\theta) = f(t, x(t_i), u(t_i), \theta) \quad (7)$$
$$y(t_i|\theta) = h(t, x(t_i), u(t_i), \theta) \quad (8)$$

5.3.2 Probabilistic Models

Models for probabilistic descriptions will be limited to the state space form. While transfer function and matrix fraction descriptions are limited to linear time-invariant systems, a state space system does not share this restriction. This form also allows the combination of a continuous system with discrete measurements (a sampled-data system) to more closely match real systems.

We cover four types of probabilistic models. The first type of model is a linear stochastic model developed by assuming a white noise approximation. The second model is a general nonlinear stochastic model. The third type is a linear Ito stochastic model based on the correct description of the noise as Brownian motion with an Ito stochastic description, and the final model is a full nonlinear Ito stochastic model.

Linear Stochastic. Linear stochastic system modeling results in the following model driven by known inputs and white noise $w(t)$ [10]:

$$\dot{x} = F(t)x(t) + G(t)u(t) + L(t)w(t) \quad (9)$$

starting from a Gaussian $x(t_0)$ with a known mean $\hat{x}_0$ and covariance P_0. Average performance can often be described by this simple stochastic differential equation sometimes referred to as Langevin's equation [11, 12].

This model is supported by a discrete (or possibly continuous) linear measurement corrupted by additive white noise $\nu(t_i)$:

$$z(t_i) = H(t_i)x(t_i) + \nu(t_i) \quad (10)$$

Since the solution of these systems is a stochastic process with many potential realizations, it is best to characterize the system by the expected value of its moments (mean, variance, etc.) The optimal (minimum mean square error, unbiased, consistent) predictor for this system is the classical Kalman-Bucy Filter.

Nonlinear Stochastic Prediction. If we want to explicitly consider system dynamics for nonlinear stochastic predictor models, there are two options: a nonlinear state space model or a simulation model. For probabilistic models, the nonlinear state space model is defined as

$$x(t_{i+1}|\theta) = f(t, x(t_i), u(t_i), w(t_i), \theta) \quad (11)$$
$$y(t_i|\theta) = h(t, x(t_i), u(t_i), \nu(t_i), \theta) \quad (12)$$

A simulation model, not to be confused with a simulation as a system description, disregards the process noise and simulates $\hat{y}(t|\theta)$ by simulating a noise free model using actual inputs and $w(t_i) = \nu(t_i) = 0$.

Ito Stochastic Prediction. As reasonable as the linear stochastic model seemed, it is not completely suitable. Although other models may be derived from these Langevin type equations, the Markovian description is typically lost. With this loss, complete knowledge of the probability density functions is required to determine system properties. This information is usually not available.

Linear stochastic differential equations can be properly developed through the use of Wiener stochastic integrals [10]. Therefore, the properly defined linear stochastic differential equation is:

$$dx(t) = F(t)x(t)dt + B(t)u(t)dt + G(t)d\beta(t) \quad (13)$$

where $\beta(\cdot,\cdot)$ is of diffusion strength $Q(t)$ for all t of interest given by $E\{d\beta(t)d\beta^T(t)\} = Q(t)dt$.

In general, characterization of this process requires the joint probability density (or distribution if the density cannot be assumed to exist) of $x(t_1), x(t_2), \ldots, x(t_N)$ for any number N of time cuts in the interval of interest by repeated application of Bayes rule. If $x(\cdot,\cdot)$ is a Markov process, however, specification of the transition probability densities completely specifies the joint densities and the transition probabilities can be propagated via the forward Kolmogorov equation.

Linear models. If the system model is linear, solution to the forward Kolmogorov equation yields the familiar form of the state and covariance update:

$$\dot{\hat{m}}_x(t) = F(t)\hat{m}_x(t) \quad (14)$$

$$\dot{P}_x(t) = F(t)P_x(t) + P_x(t)F^T(t) + G(t)Q(t)G^T(t) \tag{15}$$

(Note: In the development of error criterion, etc. derivatives must be computed using the Ito differential rule.)

Nonlinear models. If we are willing to neglect the second partial derivatives with respect to x, we can use the extended Kalman filter.

In the general case, the nonlinear problem is not solvable. There are a number of other approximations that exploit a Taylor series representation of the dynamics and measurement to estimate conditional moments. One of the more computationally reasonable is the modified Gaussian second order filter (see [10]).

5.4 Identification Methodology

We now discuss techniques for generating the estimate. A partial list of algorithms mentioned in the literature included 32 different methods, most of which can be classified by two elements: the form of the identifier and the criterion of fit. The form of the identifier defines the "experimental setup" or the manner in which the estimates are generated and compared. The criterion of fit establishes both the cost function and the method of its minimization.

Categorizing the identification method by the form and the criterion reduces the many identification methods to only five approaches: Prediction Error, Correlation, Maximum Likelihood, Optimization, and Approximation Techniques.

5.4.1 Form of the Identifier

Equation Error Method. For the equation error method, Figure 2, we use the system equations as given. Assume first that we have the following general description defined by a parameter vector $\hat{\theta}$ and that we know the form of the vector functions f and h:

$$\dot{x}(t) = f(t, x(t), u(t), w(t); \theta) \tag{16}$$

$$y(t) = h(t, x(t), u(t), \nu(t); \theta) \tag{17}$$

Now we assume that we can measure the controls, the states, and the state derivatives. With all of this information, we can determine the error between the model and the actual data: $\dot{x}_a, x_a, u_a$:

$$\varepsilon(t, \theta) = \dot{x}_a - f(x_a, u_a; \theta) \tag{18}$$

The vector $\varepsilon(t, \theta)$ is the equation errors. From these equation errors, $\varepsilon(t, \theta)$, we can form some nonnegative function such as $J(\theta) = \int_0^T \varepsilon^T(t, \theta)\varepsilon(t, \theta)dt$ and search over θ to find the minimum.

Output Error Method. The equation error method required measurement of all of the elements of the system. Often, this is not possible. The output error method is based on an output error criterion and avoids this requirement.

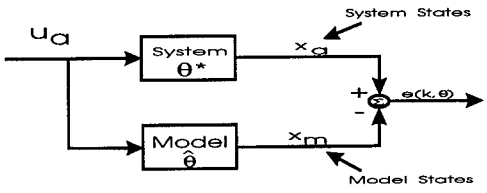


Figure 2: Equation Error Method

Prediction Error Method. The prediction error method is the third approach to developing an error function by which a parameter search can be structured (Figure 3).

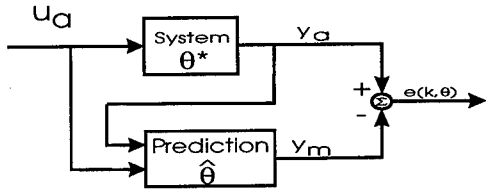


Figure 3: Prediction Error Method

Instead of comparing states or outputs, the estimated parameter, θ, is used in the model with the input u_a and the output y_a to generate an estimate of the output y_m. Given a description

$$y(t) = G(q)u(t) + H(q)e(t) \tag{19}$$

and having observed the output y and the input u, the prediction errors can be computed as

$$e(t) = H^{-1}(q)\left[(y(t) - G(q)u(t)\right] \tag{20}$$

5.4.2 Criterion of Fit

By criterion of fit, we mean the function or functional that is optimized to determine the parameter estimates.[4] We consider three criterion: minimum mean square, maximum a posteriori (maximize the CPDF), and maximum likelihood (maximize the JPDF).

Minimum Mean Square Error. Minimum mean square estimators minimize a cost function that is a function of the (possibly weighted) output error only - $J(\hat{\theta}) = \varepsilon^T W \varepsilon$. The mean square error matrix M for an estimate of $\hat{\theta}$ of θ (with b equal to the bias) is:

$$M = E\left\{(\hat{\theta} - \theta)(\hat{\theta} - \theta)^T\right\} = cov\hat{\theta} + bb^T \tag{21}$$

[4] We do not know the actual parameter vector θ_* and cannot define an error between θ^* and $\hat{\theta}$. The error must be computed from $\{z(t_i)\} \Leftrightarrow \{u(t_i)\}$ and $\{y(t_i)\}$

Both bias and covariance must both be minimized to attain the minimum mean square estimate; and, in general, the minimum m.s.e. will be biased.

Maximum A-Posteriori. The Bayesian approach to parameter estimation assumes a parameter vector with *a priori* (before the measurement) probability densities $P(\theta)$. The observations Z^N, therefore, are correlated with θ. Measurements are used to determine the most likely value after the measurement, the Maximum a posteriori (MAP) estimate $\hat{\theta}_{MAP}$ via the application of Bayes rule:

$$P(\hat{\theta}|z) = \frac{P(z|\hat{\theta}) \times P(\hat{\theta})}{P(z)} \tag{22}$$

Here $P(z|\hat{\theta})$ is the conditional probability; i.e., the total probability of the measurement conditioned on the current estimate of θ.

We can rewrite the maximization to be the minimization of the negative logarithm of $P(z|\hat{\theta})$:

$$\hat{\theta}_{MAP} = arg(\hat{\theta}) \min_{\theta} \left[-\log P(\hat{\theta}|z) \right] \tag{23}$$

where $\log P(\hat{\theta}|z) = \log P(z|\hat{\theta}) + \log P(\hat{\theta}) - \log P(z)$.

Maximum Likelihood. Given that the joint probability of the random vector to be observed is $f_z(\theta; Z^N)$, then the probability that the random variable will produce the realization Z^N is proportional to $f_z(\theta; Z^N)$. Once a particular realization Z_*^N is inserted into the joint PDF, this becomes deterministic and is called the likelihood function. A maximum likelihood estimator maximizes this function:

$$\hat{\theta}_{ML} = arg(\theta) \max_{\theta} f_y(\theta; Z^N) \tag{24}$$

so that the observed event becomes as likely as possible.

6 Results and Conclusions

This new approach provides a structured method of developing metamodels for simulations. In each case, we step through decisions that are based on existing information or follow from prior decisions. We have added the capability to explicitly model dynamical systems and defined the requirements to use these as metamodels.

We simplified the metamodeling process to two phases: problem definition and the metamodeling process. In the problem definition we begin with an analysis of the metamodel requirements and the simulation under study. We then progress to the description of the system (not the model) so that we will be able to select a metamodel structure that matches both the requirements and simulation that we are going to metamodel.

Definition of the model set for the metamodeling process was clearly defined by a system description, system class, a metamodel structure, and an identification methodology. All of these selections came directly from the problem definition.

References

[1] D. Caughlin, "A Metamodeling Approach to Model Abstraction," *Proc. 1994 Fourth Annual IEEE Dual Use Technologies and Applications Conference*, May 1994.

[2] M. A. Zeimer, et. al., "Metamodel Procedures for Air Engagement Simulation Models," *IRAE Technical Report*, Jan 1993.

[3] V. Vemuri, *Modeling of Complex Systems*, Academic Press 1978.

[4] L. Ljung, *System Identification: Theory for the User*, Prentice-Hall, New Jersey, 1987.

[5] D. Caughlin, "An Evaluation of Simulated Annealing for Modeling Air Combat Simulations," Proc. 1994 IEEE Dual-Use Technologies & Application Conference, May 1994.

[6] L.B. Anderson, et al, "SIMTAX, A Taxonomy for Warfare Simulation," Workshop report taken from the *Catalog of Wargaming and Military Simulation Models, 11th Edition*, Force Structure, Resource, and Assignment Directorate (J-8), The Joint Staff, Washington, DC 20318-8000, September 1989.

[7] J. C. Willems, "Paradigms and Puzzles in the Theory of Dynamical Systems," *IEEE Trans. on Automat. Contr.*, vol. 36, no. 3, pp. 259-294, March 1991.

[8] Sinha, Kusta, Modeling and Identification of Dynamic Systems, Van Nostrand, 1983.

[9] J. P. Norton, An Introduction to and Identification, Academic Press, 1988.

[10] P. S. Maybeck, *Stochastic Models, Estimation, and Control, Vol 2*, Academic Press, 1982.

[11] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, 1965.

[12] L. Ingber, "Statistical Mechanics of Combat and Extensions," reprint from *Toward a Science of Command, Control, and Communications*, AIAA, December 1993.

Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

AUTOMATING THE METAMODELING PROCESS

Don Caughlin

Space and Flight Systems Laboratory
University of Colorado at Colorado Springs
Colorado Springs, Colorado 80933-7150, U.S.A.

ABSTRACT

Model abstraction using metamodeling has demonstrated the capability to facilitate software reuse, large scale model integration, verification, and validation. Once restricted to static representations that represented the input-output behavior of models, research has developed the capability to build dynamic metamodels. This capability results from a new approach supported by a taxonomy of metamodeling problems, solution structures, and metamodeling methods. The development of the metamodel, however, still requires a thorough understanding of model abstraction, reduced order modeling, and system identification. In addition, even with the most robust procedures it is possible that the desired data generated by a simulation model will not meet the assumptions or numerical requirements of the procedure. Consequently, there is a requirement for a robust metamodeling support system that will support the subject matter expert. Automation of the metamodeling process will assist the analyst who is not familiar with model abstraction techniques but needs to reuse a piece of code, integrate different models, or verify a new version of a simulation. This paper describes the design of a Metamodeling Support System that provides this automation.

1 INTRODUCTION

A metamodel is a mathematical approximation of the system relationships defined by a more detailed model (Caughlin et al. 1997a). Caughlin (1995) introduced a structured approach to metamodeling that separated the procedure into two steps: problem definition and an iterative metamodeling process. While we can generate a metamodel from data generated by any model structure, the discussion in this paper is limited to metamodels of simulations.

We defined a metamodeling problem as the direct sum of the metamodel requirements and the model (simulation) to be approximated. To support this definition, the problem definition step first determines the purpose of the metamodel. In the definition of this purpose we identify the input and response that we are interested in and determine the important characteristics of these data. Also for this purpose, we define the region of interest, validity measures and specify the required validity. In addition to metamodel requirements, problem definition addresses the second part of the direct sum and characterizes the simulation that is the subject of the metamodel. This characterization provides data that can be used to match the simulation's characteristics to the metamodel structure and identification method.

The second portion of the structured approach was an iterative metamodeling process which consists of the following steps (Caughlin 1997c):

1. Select an Experimental Design
2. Run the Simulation
3. Collect Data
4. Select a Metamodel Set
5. Select Identification Methodology
6. Generate the Metamodel

This approach supported development of dynamic metamodels that exhibit memory and can model phenomena where the past influences the future. In addition, a more robust identification procedure was developed that could be applied to a broader range of problems than existing techniques.

The revised process outlined above provides a direct method of sorting through the myriad of decisions necessary to develop a dynamic metamodel and reduces the number of independent decisions required to develop the metamodel. This process is supported by a new taxonomy of problems, structures, and methods and set of computer aided routines that match the problem definition with the simulation characteristics.

Even with a new approach supported by a taxonomy of metamodeling problems, solution structures, and metamodeling methods, the development of the metamodel still required a thorough understanding of model abstraction, reduced order modeling, and system identification.

In addition, even with the most robust procedures it is possible that the desired data generated by a simulation will not meet the assumptions or numerical requirements of the identification procedure.

Consequently, the widespread use of metamodeling as a method of model abstraction requires an automated support system to assist the analyst. This paper describes research into the design of a prototype Metamodeling Support System (MSS) to automate model abstraction. The prototype system will assist the analyst who is not familiar with model abstraction techniques but needs to reuse a piece of code, integrate different models, or verify a new version of a simulation.

2- DEVELOPMENT AND SYSTEM OVERVIEW

The MSS program provides a semi-automated support system to assist an analyst/modeler in developing a metamodeling abstraction of a more detailed model. This system supports the metamodeling approach outlined above and covered in the references.

2.1- Technical Program

The MSS development program is divided into two development phases:

1. Build 1
2. Prototype Metamodeling Support System

The MSS Build 1 establishes the baseline and provides the following capabilities:

1. A metamodeling system based on an object-oriented architecture that is capable of future expansion.
2. The capability to analyze the source code, generate and run the simulation, and gather data.
3. Data storage and analysis routines.
4. Metamodeling routines and procedures to generate and verify the metamodel.

In Build 1, the MSS provides an executive and automated routines to analyze and run the simulations to gather the data for the metamodel. Existing identification algorithms will be incorporated into this system to provide the basic capability to generate metamodels.

The automated system support discussed above will be provided by an expert system. An expert system is the union of declarative knowledge and inference. The knowledge base contains the declarative knowledge. The inference engine controls the application of that knowledge. It is an algorithm that dynamically directs or controls the system when it searches the knowledge base.

The Prototype Metamodeling Support System is a near-term upgrade of the basic Build 1 and adds the following capabilities:

1. An expert system.
2. Supporting Knowledge Base to support decisions required to develop metamodels.

Documentation for the program is provided in a System/Subsystem Specification (SSS), System/Subsystem Design description (S/SDD), System Software Design Description (SDD).

2.2- System Capabilities

The system must provide the general housekeeping, expert system, and knowledge base to support the objectives and decisions outlined in the metamodeling approach shown in Table 1.

There are four general capabilities that must be provided by this system. These areas are the analysis of the simulation, the correlation of the simulation and data with a metamodel structure and identification method, generation of the metamodel, and finally, the analysis of the metamodel.

First, the system needs to handle the general housekeeping associated with any experimental setup such as: user preferences; cataloging the input and output data; associating the data with parameter selections; and tracking the status of the metamodeling session.

Secondly, the system needs to support problem definition. This includes definition of the metamodel purpose and the analysis of the simulation.

Once the metamodeling problem has been defined, the system must support selection of a metamodel representation (structure) and method of identification. Given a structure and method, the system must now parameterize the metamodel. Lastly, the system should support the analysis of the metamodel. These capabilities are covered under the metamodeling process.

Specific, more detailed, requirements to support these capabilities were provided in a Statement of Work, previous research, and an analysis of current trends in model abstraction.

Table 1: Metamodeling Approach

MAJOR AREA	OBJECTIVE	DECISION/ACTION
Problem Definition	Metamodel Purpose	Scope Use
	Simulation Characteristics	External Characteristics Internal Characteristics
Metamodeling Process	System Representation Identification Methodology	System description
		System class
		Metamodel Structure
		Identification Methodology
	Generate and verify metamodel	Experimental Design
		Run the Simulation
		Collect Data
		Generate the Metamodel
		Verify the Metamodel

3 DESIGN PROCESS

The fact that the MSS must interact with a variety of different legacy simulations with unknown structure dictates a robust, modular, scaleable, and extensible design. A point design would not be able to adapt to different model or simulation structures or handle the different types of analysis to be performed. This dictate, and the fact that this was a software development, seemed to demand an Object-Oriented design approach.

System capabilities, however, stem from the requirement to support a structured sequential process. The functionality is process related and does not reside in or be derived from any of the objects that exist in the environment. Also, the MSS is not a component of another system but a system of systems under the supervisory control of the MSS executive. This analysis supports a structured Systems Engineering design approach.

While Systems Engineering provides a high-level functional architecture, Object-Oriented (OO) Modeling and Design generates a set of lower level functions that should (more properly) be called methods or operations. Unfortunately, it is usually not possible to distribute the methods of the OO classes among the different functional elements that result from Systems Engineering. Consequently, at this point there are two incompatible structures. This issue was addressed in Caughlin (1997b). The design of the MSS followed the method proposed in that paper. A summary of the method follows:

1. Follow the standard Systems Engineering process generating the system capabilities with a functional decomposition and allocation of requirements.
2. Initiate the Object-Oriented Modeling and Design process identifying the underlying objects that are basic to the problem at hand. Identify object attributes, operations (methods), relationships and associations. Develop a class structure, prototype code, and data dictionary.
3. Beginning with system capabilities (requirements), define operating "States and Modes" of the system that are consistent with the functional architecture. Display these states and modes in a flow chart.
4. Using the functional capabilities (architecture) and the States and Modes Flow Chart, connect the functionality that comes from the Systems Engineering process to the objects that result from Object Modeling Techniques by the definition of abstract "manager" and "controller" objects that connected the "top down" functionality with the "bottom up" objects.

4 SYSTEM DESIGN

Presentation of the design of the MSS is organized under Requirements Analysis, Functional Design,

Object-Oriented Design, and KnowledgBase Design. This section concludes with the resulting System Architecture. Requirements Analysis and Functional Design followed the standard Systems Engineering Process (EIA/IS-632 1994).

4.1 Requirements Analysis

An analysis of the required functionality and the process that the MSS is to support led to 511 requirements. Requirements Traceability and Management was accomplished with a CASE tool – Requisite Version 2.0.18.

System Capabilities were organized as follows:

1. Interface Capabilities
   (a) User Login
   (b) Session Establish/Restore
   (c) Session Configure
   (d) Select Operation
2. Problem Definition Capabilities
   (a) Metamodel Purpose
   (b) Simulation Characteristics
3. Metamodel Capabilities
   (a) Select Metamodel Set
   (b) Select Identification Method
   (c) Select Experimental Design
   (d) Run Model
   (e) Fit Metamodel
   (f) Verify Metamodel

Additional (nonfunctional) capabilities and constraints were also identified. Internal and external interfaces were defined.

4.2 Functional Design

System capabilities were decomposed and allocated to functions based on the following required States and Modes: Login (Standby); Configure (Define) Session; Problem Definition; Metamodel; and Maintenance States. The operating modes are "Manual," "Assisted," and "Automatic" and apply primarily to the Problem Definition and Metamodeling States. These modes determine the level of support provided by the Expert System.

4.2.1 States and Modes

Login (Standby) State. This is the initial state of the system prior to login to the MSS. In this mode, the system will determine who the analyst is, which process is to be modeled, and the status of the process at login. This state allows the analyst to suspend a session and come back to it at a later time. This state operates only in the manual mode although defaults are provided.

Maintenance State. This state allows the various maintenance functions. This state supports file and knowledgebase maintenance. In addition, user profiles and preferences are established in the maintenance state. Again, this state operates only in the manual mode.

Configure Session State. In this state, the analyst defines the objectives of the session. Here we identify the simulation that will be modeled and provide the data that will support the Problem Definition State. This state can operate in both the manual and assisted mode and cannot be exited until all of the data is provided.

Problem Definition State. The Problem Definition State can function in both the manual and assisted modes. This state provides all of the data defined as *a priori* information. There are two major areas that are addressed. The first area is the purpose of the metamodel. The second area is the characteristics of the simulation that is to be metamodeled.

Metamodel State. The Metamodel State provides the ability to complete the metamodeling procedure. These steps include selection of the metamodel set and identification method, selection of the experimental design, running of the model, fitting the metamodel, and finally, verification of the metamodel. This state operates in all modes.

4.2.2 Functions

Analyzing the Required Capabilities with respect to the States and Modes resulted in the following functions for the requirements allocation.

User Interface (UI). The UI component provides the multimedia control and display interface to the user. It interprets and error checks user inputs and it provides graphical, text and video displays, and audible alarms. It displays out-of-tolerance conditions visually and, if it is a critical parameter, audibly.

Data Manager (DM). The DM provides all of the functionality associated with the management of MSS data. As such, the DM supports data requests from all other functions. Data archiving is accomplished on a Load/Save basis as opposed to a data entry basis.

Scenario Manager (SM). The purpose of the SM is to structure and manage the data used to generate the metamodel. The SM provides three different types of support to the MSS.

First, the Scenario Manager supports data gathering for the problem definition steps of the process. At this stage, the Scenario Manager determines prior information for construction of the metamodel.

Next, the Scenario Manager uses the data from problem definition to generate input data for the simulation. The combination of simulation input and output becomes the input for the identification routines that generate the metamodel. The SM manages the input and output data (through the Data Manager) that will be used to generate a metamodel.

Lastly, the Scenario Manager provides the ability to link the various functions to complete the metamodeling process. Development of the metamodel is a multi-stage process. In the first stage we determine the purpose and characteristics of the simulation. Complete determination of the simulation characteristics, however, requires the output data from the simulation which is provided by the Metamodel Manager. Consequently, the process moves from the Scenario Manager to the Metamodel Manager and back to the Scenario Manager. The last type of support provides by the SM is in tracking this interaction.

These stages are iterative and the sequence of the operation can vary depending on the data and the outcome. Based on data from the Problem Status File generated by the Session Manager (discussed below), the Scenario Manager first determines the status of the solution and what data is required to proceed to the next step of the metamodeling process. From this analysis, the proper SM response is selected.

Metamodel Manager (MM). The MM provides the capability to generate the metamodel.

The first step is to postulate a metamodel. The MM assists with the initial definition of the metamodel structure and guides the selection of the metamodel set. The MM should provide a recommended metamodel set based on the problem and the simulation that is to be metamodeled.

Given the metamodel set, the next decision is the selection of the ID methodology. When we have established the metamodel set we should compare the metamodel set to the metamodeling problem to insure consistency of the metamodel problem. With the metamodel set and ID methodology determined, we use this information to define requirements for the Experimental Design. These selections constrain the Experimental Design and define the Input-Output Data requirements.

The MM subsystem should provide a recommended experimental design based on the problem definition, metamodel set, and ID methodology.

Once the experimental design is defined it should also be compared to the metamodel problem to insure that the design and problem are consistent. Based on the metamodel set, ID methodology, and Experimental Design, we can identify appropriate analysis tools. This step also identifies preprocessing data analysis required to verify the results of the design.

Once the metamodel set, ID methodology, Experimental Design, and analysis tools are defined the simulation controller can configure data capture files and then run the simulation to generate the output data. The I/O must be configured for each simulation along with the simulation run times and message passing. At this time we load simulation and configuration files and execute the simulations as defined.

This data must be analyzed (before the generation of the metamodel) to insure that it meets the restrictions of the method (Belsley 1980, Ljung 1987). In general, we:

1. Assess for collinearity
2. Remove trends and Outliers
3. Select useful portions
4. Filter to enhance important frequency ranges

The MM now gets metamodel data files and metamodel parameters using the data manager. With the data from the simulation, the metamodel set and the ID methodology, the MM now fits the metamodel to the data. After generation of the metamodel the MM then must verify that the metamodel meets the requirements of the problem definition.

Session Manager (SEM). The SEM manages the status of the current session. First the SEM must identify the user and their status as Expert / Advanced / Novice. The SEM then gets a general idea of what the objective of the session. From these objectives the SEM determines if special resources are required.

The SEM also provides the ability to suspend sessions, recover from, and continue with a previous session if requested.

The SEM then configures and manages the session and the session state via the Login and Session State Files.

We have discussed capabilities (requirements) and a functional decomposition and allocation that meets these requirements. Rather than proceeding with the

design process, our methodology dictates an Object-Oriented approach to the problem. We discuss this analysis next.

4.3- Object-Oriented Design

Beginning with the same problem statement, application of Object-Oriented Modeling and Design to the requirements results in the following primary objects:

1. Analyst
2. Project
3. Problem
4. Simulation
5. Metamodel
6. Metamodel Set
7. Metamodel Parameters
8. Data

This analysis continued with identification of object attributes, operations (methods), relationships and associations. A class structure, prototype code, and data dictionary was developed. Object-Oriented Modeling and Design was supported by OMTool that was developed by General Electric Advanced Concepts Center and implements Object-Oriented Modeling and Design as defined by Rumbaugh (Rumbaugh 1991).

In a typical OO methodology, the next Analysis step is to develop the dynamic model by preparing scenarios of typical interaction sequences, identifying events that occur between objects, preparing an event trace for each scenario and an event flow diagram for the system. A functional model is used to describe the transformation from input to output by determining input and output values and developing data flow diagrams to show functional dependencies and identify constraints.

In the design methodology that we follow, however, this data is provided by the Systems Engineering process. We do not continue with the OO design but use these object classes to populate the "player" or lower level of the architecture.

Manager objects are defined that implement the functionality defined by the system states and modes. The player level identifies the objects that must be addressed, Intermediate level "controller" objects are designed to make the connections between the manager and player levels.

4.4- KnowledgBase Design

Expert System support is provided for two purposes. The first purpose is to assist in execution of the Metamodeling process as we have defined it. The process can be executed using many different sequences. The Expert System constrains this sequence to insure that required information is available at each step and that results conform to assumptions.

The second area of support is assistance in decisions required by the process. Here, we help with selection of the metamodel structure, identification method, analysis tools, etc. This is the Decision Assistance Knowledgebase.

The Metamodeling Process Knowledgebase is part of the original specification since it's contents are well known. The MSS contains the ability to record and incorporate the metamodeling results. The Decision Assistance Knowledgebase will be developed as the MSS is used to generate metamodels by recording decisions and the effectiveness of these decisions.

4.5- System Architecture

The software architecture is a framework for the interconnection of subsystems within some major system - in this case the MSS. Each of these component systems are defined by their capabilities and are composed of functions (subsystems) which in turn are a collection of objects (modules).

The MMS is composed of six levels: the top level, the manager level, the component level, the player level, the data level, and the library level (only 4 are shown in Figure 1). The top level encapsulates the abstraction of the MMS and supports the four sequential processing steps: system login, configure the session, define the metamodeling problem, generate and verify the metamodel. The ability to maintain the system is also provided. This level is described by the "states and modes" of operation.

This functionality is implemented by subsystems derived from the functional allocation by objects of the "manager class." This class is expected to completely support MMS capability requirements in these five processing steps. This class of components are instantiated as the different objects required to provide this functionality. These objects are the User Interface (UI), Data Manager (DM), Scenario Manager (SM), Metamodel Manager (MM), and the Session Manager (SEM).

The lower "player" level consists of the objects that are generated by the Object-Oriented Design. The player level encapsulates the entity object classes that are the inputs and products of the MSS such as the

simulations, metamodeling problems, and metamodels. These are the entities that will be generated and/or manipulated in the course of the generation of the metamodel.

The connection between the manager and player levels is accomplished by the definition of an intermediate level. This intermediate level is the collection of subsystems that perform the various operations of the MSS. In the description of the system, they are called "controllers." The component level "controllers" provide the connection between the managers and player objects.

Table 2 below shows the "Manager" class, the Systems Engineering functions performed by the class and the objects of the OO design from OMT that are affected.

A data level consists of the data objects required to manage the processes and control the products. Lower level library objects also exist that are used to implement standard functions that are not explicitly named.

5- IMPLEMENTATION

Many of the components required to meet the functional requirements of the MMS already exist.

An existing code analyzer can be used to analyze the simulation characteristics. Data can be efficiently stored in any number of relational databases (e.g. external simulation characteristics are already provided in a SIMTAX database (Anderson, et al. 1989).

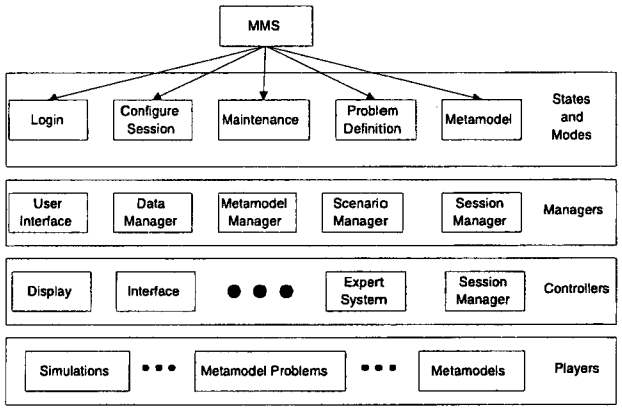


Figure 1: MMS Architecture

In addition, there are a number of expert systems that could be used to provide automation support. Identification and analysis routines are available as well as a number of numerical engines.

Rather than develop all of the components of the MSS, the decision was made to develop a shell or mainframe that would integrate and manage both new and existing components.

This shell was developed with Microsoft Development Studio and C++ language using the Microsoft Component Object Model, the Microsoft Foundation Class Library, and the DAO database interface. MSS targets the Windows NT operating system.

The OO design was accomplished in OMTool. These files will be integrated into the Microsoft Development Studio.

Table 2: Connection Between Objects and Functions

MMS MANAGER	FUNCTIONALITY (SE)	PLAYER (OMT)
User Interface	Interface	Analyst
Data Manager	Interface	Data
Scenario Manager	Problem Definition	Problem Simulation
Metamodel Manager	Metamodel	Metamodel Metamodel Set Metamodel Parameters
Session Manager	Interface	Project

The Expert system is provided by the C Language Integrated Production System (CLIPS) developed by the Software Technology Branch (STB), NASA/Lyndon B. Johnson Space Center. CLIPS is designed to facilitate the development of software to model human knowledge or expertise. Rules and objects form an integrated system since rules can pattern-match on facts and objects. In addition to being used as a stand-alone tool, CLIPS can be called from a procedural language, perform its function, and then return control back to the calling program.

CLIPS was embedded into the MSS as a DLL using a Wrapper Class provided by Mark Tomlinson (MTOMLINS@us.oracle.com).

The numerical engine is provided by MATLAB. Identification and analysis tools are incorporated as MATLAB "M" files.

Documentation for the system is provided in the form of Windows help files which are assessable online.

6- SUMMARY

This paper has described the design and capabilities of a prototype Metamodeling Support System that will assist the analyst who is not familiar with model abstraction techniques but needs to reuse a piece of code, integrate different models, or verify a new version of a simulation.

We presented an outline of the capabilities required to support the Metamodeling process and references where details may be found.

We demonstrated the use of a design process that integrated Systems Engineering and Object-Oriented Modeling and Design to provide a system architecture that meets functional requirements and accommodates an Object-Oriented framework.

Unfortunately, the scope of the paper does not allow a complete description of implementation details. A summary was provided.

ACKNOWLEDGMENTS

This research was partially supported by Rome Laboratory under Contract No. F30602-96-C-0040/P0001.

REFERENCES

Anderson, L. B., et al. 1989. SIMTAX, A Taxonomy for Warfare Simulation, Workshop report taken from the *Catalog of Wargaming and Military Simulation Models, 11th Edition*, Force Structure, Resource, and Assignment Directorate (J-8), The Joint Staff, Washington, DC 20318-8000.

Belsley, D, E. Kuh, R. Welsch. 1980. *Regression Diagnostics.* New York:John Wiley & Sons.

Caughlin, D. 1995. *Final Report, Modeling Techniques and Applications, Volume I.* USAF Contract F30602-94-0110, Rome Laboratory/IRAE, 32 Hangar Rd, Griffis AFB, NY 13441-4114.

Caughlin, D.,A. F. Sisti. 1997a. "A Summary of Model Abstraction Techniques". In *Enabling Technology for Simulation Science, Alex. F. Sisti Editor, Proceedings of the SPIE*, Vol. 3083:14–21.

Caughlin, D. 1997b. "Integration of Object-Oriented and Functional Modeling and Design Methods." In *Enabling Technology for Simulation Science, Alex. F. Sisti Editor, Proceedings of the SPIE*, Vol. 3083:89–99.

Caughlin, D. 1997c. "Model Abstraction Via Solution of the Inverse Problem to Define a Reduced Order Model". Accepted for publication in *SCS Transactions*

EIA/IS-632. 1994. *EIA Interim Standard, Systems Engineering.* Electronic Industries Association.

Ljung, L. 1987. *System Identification: Theory for the User*, New Jersey:Prentice-Hall.

Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. 1991. *Object-Oriented Modeling And Design.* New Jersey: Prentice Hall.

AUTHOR BIOGRAPHY

DON CAUGHLIN is Acting Director of the Space and Flight Systems Laboratory at The University of Colorado at Colorado Springs. He received a BS in Physics from the Air Force Academy, an MBA from the University of Utah and MS and Ph.D. degrees in Electrical Engineering from the University of Florida. His research interests include modeling and simulation, system identification, pattern recognition and intelligent control. Dr. Caughlin has over 28 years experience as an experimental test pilot, chief scientist, research scientist, program manager and was also Associate Dean of the School of Engineering at the Air Force Institute of Technology. He is a senior member of IEEE and AIAA and a member of SCS and the Society of Experimental Test Pilots.

*U.S. GOVERNMENT PRINTING OFFICE: 2000-510-079-81237